SH-Elektronik
Soft- und Hardware

GAL DEVELOPMENT SYSTEM

G
D
S ***Version 3.5***

Benutzer Handbuch

© SH-Elektronik 1996 - 2001

Lizenzvereinbarung

Sämtliche Produkte von SH-Elektronik werden mit einer Lizenz zur Anwendung, wie untenstehend ausgeführt, an den Erstkäufer verkauft. Alle weiteren Rechte verbleiben bei
SH-Elektronik. Bei einer eventuellen Weitergabe gilt diese Lizenznahme automatisch mit dem neuen Lizenznehmer als vereinbart.

Erlaubte Benutzung

Die Software kann auf jedem kompatiblen Rechner zum Einsatz kommen, den Sie besitzen oder benutzen. Die Runtimelizenz beinhaltet sowohl privaten als auch geschäftlichen Einsatz der Software. Voraussetzung ist in jedem Fall, daß die Software zu jedem Zeitpunkt nur auf einem Rechner installiert und in Gebrauch ist.

Nicht erlaubte Benutzung

Sie dürfen nicht:

1. Die Benutzung dieser Software in einem Computer Service Business, Netzwerk, Timesharingbetrieb, Multi-CPU oder Multiuser-Umgebung anbieten, falls nicht jeder Benutzer mit einer individuellen Runtimelizenz seitens SH-Elektronik ausgestattet ist;

2. Kopien der Dokumentation von SH-Elektronik und der Programmdisketten anfertigen (außer zur Erstellung von Backup-Kopien);

3. Änderungen in der Software vornehmen;

4. Unterlizenzen, Leasing oder andere Rechte an dieser Software an Dritte anbieten;

5. verbale oder mediale Übersetzungen der Dokumentation anfertigen;

6. Anpassungen der Software zur Benutzung auf nicht kompatibler Hardware vornehmen;

7. Datenfernübertragung dieser Software vornehmen.

Gewährleistung

SH-Elektronik bietet eine Gewährleistung für eine Periode von neunzig Tagen, beginnend mit dem Tag der Auslieferung, dahingehend, daß die Disketten, auf denen das Programm geliefert wird, bei normalem Gebrauch in Bezug auf Material und Verarbei-tung einwandfrei sind und daß das Programm bei normalem Einsatz und ohne unauto-risierte Modifikationen ohne ernsthafte Fehler, die es unbrauchbar machen würden, abläuft. Die Garantie unterliegt der Voraussetzung, daß das Programm unter Vorlage einer Quittungskopie an SH-Elektronik oder an einen Vertragshändler zurückgegeben wird. Die Haftung von SH-Elektronik sowie die dem Käufer zur Verfügung stehenden Rechtsmittel im Rahmen dieser Garantie beschränken sich ausschließlich darauf, daß SH-Elektronik nach freiem Ermessen versuchen kann, Fehler zu berichtigen bzw. mit Maßnahmen, die SH-Elektronik zur Behebung des jeweiligen Problems geeignet erscheinen, bei der Fehlerberichtigung behilflich zu sein, oder das Programm oder einzelne Disketten durch funktionsmäßig gleichartige Software oder Disketten zu ersetzen oder den Kaufpreis zu erstatten und diesen Vertrag zu beenden.

Alle über die vorstehenden ausdrücklich aufgeführten begrenzten Garantie-bestimmungen und -bedingungen hinausgehenden Gewährleistungen seitens SH-Elektronik oder Ansprüche des Käufers sind ausgeschlossen, seien sie ausdrücklich oder stillschweigend miteingeschlossen, gesetzlich vorgeschrieben oder im Rahmen gegenseitiger Mitteilungen gegeben. Insbesondere ist eine stillschweigend miteingeschlossene Garantie durchschnittlicher Qualität oder der Eignung für einen bestimmten Zweck ausgeschlossen.

SH-Elektronik gewährleistet nicht, daß das Programm ohne Unterbrechungen oder Fehler abläuft.

Haftungsbeschränkung

Unter keinen Umständen haftet SH-Elektronik für irgendwelche Schäden, einschließlich Datenverlust, entgangenem Gewinn, Deckungsbeiträgen oder anderen Neben-, Folge- oder indirekten Schäden, die sich aus der Benutzung des Programms oder der beiliegenden Dokumentation ergeben können, und zwar ungeachtet der Schadensursache oder der Grundlage des Haftungsanspruchs. Diese Beschränkung gilt sogar dann, wenn SH-Elektronik oder ein Vertragshändler auf die Möglichkeit eines solchen Schadens hingewiesen worden sind. Der Käufer bestätigt, daß die Lizenzgebühr dieses Risiko einschließt.

Hinweise

* Alle Rechte vorbehalten. Reproduktionen, auch von Teilen, dieses Manuals sind in jeglicher Form ohne die schriftliche Genehmigung von SH-Elektronik untersagt.

* Der Inhalt dieser Bedienungsanleitung kann ohne besondere Ankündigung fortgeschrieben werden.

* Das vorliegende Manual wurde sorgfältig bearbeitet. Sollten dennoch Fehler oder Inkonsistenzen auftreten, bittet SH-Elektronik um entsprechende Hinweise.

* Trotz der sorgfältigen Bearbeitung kann keine Gewährleistung für Fehler in dieser Bedienungsanleitung und deren Auswirkungen übernommen werden.

Copyright 1996 - 99 by:
SH-Elektronik

IBM, PC, PC/XT, PC/AT sind registrierte Schutzmarken der International Business Machines Corp.

MS-DOS, OS/2 sind registrierte Schutzmarken der Microsoft Corp.

DESQview, DESQview386 sind registrierte Schutzmarken der Quarterdeck Office Systems

PAL, PALASM sind registrierte Schutzmarken der Monolithic Memories Inc.

ispGAL, GAL sind registrierte Schutzmarken der Lattice Semiconductor Inc.

Inhaltsverzeichnis

Kapitel 1: **Einführung 6**

Kapitel 2: **Programminstallation 7**

Kapitel 3: **Die Benutzung von GDS 9**

3.1 Compilierung via GDS 9
3.2 Simulationen des Source- und JEDEC-Codes 12
3.3 Programmierung eines GAL..... 15
3.4 Die Einstellung im Menü Optionen..... 17

Kapitel 4: **Einführung in die GAL-Assembler Syntax 20**

Kapitel 5: **Details über GALs 25**

5.1 Die GAL16V8-Familie 25
5.2 Die GAL22V10-Familie 32
5.3 Die GAL20RA10-Familie 40
5.4 Der ispGAL22V10 44
5.5 Die GDSxx-Familie 45

Anhang A**: Fehler- und Warnungsmeldungen**

Anhang B**: Sourcecode der Beispielfiles**

Anhang C**: Überblick der Hotkeys und Editorkommandos**

Anhang D**: Übersicht der programmierbaren GALs**

Einführung

Mit dem Beginn der Entwicklung von digitalen Layouts und Platinen begann auch die Benutzung spezieller integrierten Schaltungen, in Besonderheit von SSI- und MSI-Logikbausteinen. Die bekannteste IC-Serie dürfte dabei die 74xxx-Familie sein, die eine Vielzahl von Funktionalitäten innerhalb der gesamten Serie anbietet, wobei jedoch die einzelnen Bausteine fest verdrahtet sind.

Eine Folge der frühen Entwicklungsphase war die Einführung von Mikroprozessoren an Stelle von kundenspezifischen ICs oder Layouts mit dem Vorteil der Software-kontrollierten Funktionalität. Flexibilität auf der einen Seite, geringe Geschwindigkeit (im Vergleich mit TTL-Schaltungen) auf der anderen waren das Ergebnis, so daß es nur logisch war, daß ca 1 Jahrzehnt später erneut der Trend zu anwenderspezifischen ICs (ASIC) ging.

ASICs waren und sind nach wie vor eine Lösung für spezialisierte Schaltkreise mit großer Auflage. Die Nachteile für den eigentlichen Benutzer lagen jedoch nicht nur in der Unflexibilität nach Produktion, sondern auch in der Offenlegung seines Know-Hows Dritten gegenüber. So war der nächste Schritt, die Einführung halbfertigen Bausteine mit Programmiermöglichkeit beim Anwender ("Feld-Programmierbarkeit") der folgerichtige Schritt.

Innerhalb des Gefüges der feldprogrammierbaren ICs spielen PALs (Programmable Array Logics) und - später - GALs (Generic Array Logics) eine wichtige Rolle. Man kann sie zusammenfassend als monolithische PLDs (Programmable Logic Devices) oder Simple PLDs bezeichnen, wobei sie eine Reihe von Anwendungsvorteilen bieten:

- geringe Kosten, leichte Handhabung
- große Flexibilität (Ersparung von "TTL-Gräbern")
- sehr geringe Entwicklungszeiten im Minuten- und Stundenbereich

Letzterer Vorteil wird durch Ihr neues GAL Development System noch wesentlich verstärkt, da in einer benutzerfreundlichen, SAA-kompatiblen Oberfläche alles von der Sourcecodeerstellung, über die Assemblierung und Simulation bis hin zur Programmierung integriert ist, externe Programmaufrufe z.B. für andere Programmiergeräte eingeschlossen.

Diese Bedienungsanleitung bietet Ihnen sowohl eine Einführung in die Benutzung von GDS (Kapitel 2 und 3) sowie Informationen über die Programmierung in der Assemblersyntax (Kapitel 4) und über die interne Struktur der GALs (Kapitel 5).

Programminstallation

GDS wurde speziell für den Einsatz auf Personal Computer konzipiert. Um die aktuelle Version 3.5 starten zu können, benötigen Sie einen IBM PC, PC/XT oder IBM PC/AT oder einen entsprechend 100% kompatiblen Rechner mit einem minimalen Speicherausbau von 640 kByte. Die Bedienung von GDS geschieht via Tastatur oder Maus.
Die physikalische Programmierung der GALs ist in GDS Version 3.5 für die GAL-Typen 16V8, 20V8, 18V10, 22V10, 26CV12 und 20RA10, (jeweils A,B,D Typen), die ispGAL-Typen ispGAL22V10, GDS14, GDS18 und GDS22 sowie die PALCEs 16V8 und 22V10 implementiert. Benutzen Sie als Programmiergerät für die GALs bitte den in mc 3/93 vorgestellten GDS-Programmer mit den entsprechenden Adaptersockeln, für die GALs und PALCEs den neuen Programmer GDSProg2, sowie den hed.chip, für die ispGALs das spezielle isp-Kabel, oder generell ein Programmiergerät, das JEDEC-kompatible Dateien verarbeitet.

Die Verbindung zum Programmer wird via einem LPTx- oder einem entsprechenden Parallel-Port aufgenommen. Die Basisadresse dieses Ports kann im Menü OPTION eingestellt werden.

Für andere Programmiergeräte bietet GDS V3.5 Ihnen die Möglichkeit, bis zu 5 externe Programme, aus der Bedienungsoberfläche heraus aufrufbar, einzubinden. Sollten Sie also GDS zusammen mit einem anderen Programmiergerät benutzen wollen, so müssen Sie lediglich eine Software des jeweiligen Herstellers in der MS-DOS-Version zur Verfügung haben. Nähere Informationen entnehmen Sie bitte dem Abschnitt 3.4 OPTION.

Das Standard-Betriebssystem ist MS-DOS Version 3.1 oder höher. Netzwerkbetrieb ist grundsätzlich möglich. Bitte beachten Sie, daß GDS als single-user System konzipiert wurde und somit keine Schutzmechanismen bei gleichzeitigem Zugriff mehrerer Nutzer auf eine Sourcecodedatei bietet, wie es bei Netzwerkbetrieb grundsätzlich vorkommen kann. Der direkte Zugriff auf den Programmierport muß dem Programm gestattet werden.

Zum Netzwerk- oder Multiuserbetrieb beachten Sie bitte die Lizenzvereinbarung; Für Mehrfachlizenzen wenden Sie sich bitte an SH-Elektronik, Kiel.

Ferner sind Multitasking-Systeme und -zusätze problematisch, falls programmiert werden soll, da die Programmierung der GALs zeitkritisch ist und in gewissen Bereichen nicht unterbrochen werden darf.

Die Auslieferung von GDS erfolgt auf einer 3,5 Zoll Diskette mit 1,44 MByte Kapazität . Sie enthält die vollständige Software als ausführbaren Code mit deutschen oder englischen Menütexten, den Hilfetexten in Deutsch sowie Zusatzprogramme und Beispieldateien.

Die Installation von GDS ist unter MS-DOS sehr einfach. Sie sollten sich ein Verzeichnis einrichten oder aussuchen, in das Sie den Inhalt der Diskette kopieren .(z.B. c:\GDS).

In jedem Fall müssen die Files GDS35D.EXE, GDS_GD35D.DEF und GDS_35.HLP in das gewünschte Verzeichnis kopiert werden, alle anderen Dateien können Sie fortlassen. Alternativ zu GDS35D.EXE müssen Sie GDS35DN.EXE benutzen, falls Sie ein Programmiergerät GDSProg2 erhalten haben. Diese Version arbeitet korrekt mit dem neuen Programmierer zusammen. Alternativ können Sie auch die englischsprachige Version GDS35E.EXE installieren.

Damit ist die DOS-Installation beendet, Sie können Ihr GDS mittels des Kommandos

GDS35D <RETURN>. starten.

Nach dem Start erkennt GDS automatisch den Bildschirmadapter und wählt die schnellste Methode aus, die Bildschirminformationen darzustellen. GDS sucht ferner den Initialisierungsfile "GDSV35. INI" in dem jeweils aktuellen Directory, in dem einige Informationen der Grundeinstellung stehen. Sollte diese Datei nicht vorhanden sein, werden Defaultwerte vorgegeben. Weitere Informationen erhalten Sie im Kapitel 3.4 OPTION.

Der minimale Speicherausbau für GDS sollte 640 kByte betragen. Pro Editorfenster stellt GDS maximal 64 kByte zur Editierung bereit.

Hinweis:
Da im GDS sehr viele speicherhungrige Applikationen vorhanden sind, achten Sie bitte darauf, möglichst wenig Fenster in der Oberfläche geöffnet zu haben.

Der Programmer wird direkt in LPT1 bis LPT3 (wie im BIOS eingetragen) oder einer 100%-kompatiblen Schnittstellen eingesteckt, die Adressauswahl erfolgt unter dem Menüpunkt OPTION. Der Programmer benötigt dabei seine eigene Spannungsversorgung, wie sie z.B. für GDS-Prog mittels eines Steckernetztteiles vorgeschlagen ist.

Bei anderen Programmern entnehmen Sie bitte Einzelheiten zur Installation des Programmiergeräts und der Software den jeweiligen Anleitungen. Das Einbinden der entsprechenden Software wird im Abschnitt 3.4 OPTION erläutert.

Die Benutzung von GDS

GDS wurde als Hardwareentwicklungssystem konzipiert, somit also zur Synthese von Schaltungen auf GALs entworfen. Dies bedeutet, daß GDS Module zur Editierung, zur Assemblierung einschließlich des einfachen Syntaxcheck, zur Simulation und als Interface zu Programmiergeräte beinhaltet.

Während die Inhalte dieser einzelnen Teile GDS-spezifisch sind, ist es die Bedienung nicht, da GDS sich in der vorliegenden Version 3.5 sehr eng an die üblichen SAA-Spezifikationen hält.

GDS läßt sich dadurch in allen Teilen via Maus oder Tastatur bedienen. Zu speziellen Fragen der Bedienung gibt es ein kontextsensitives Online-Hilfesystem, das Sie zu jedem Zeitpunkt mittels der Taste <F1> aufrufen können. In diesem Hilfesystem sind Hinweise zu der jeweiligen Aktion, bei der Sie es aufgerufen haben vorhanden; Querverweise, die unterlegt bzw. hervorgehoben dargestellt sind, erreichen Sie mittels der Maus durch zweimaliges Anklicken mit der linken Maustaste oder durch die Tabulatortasten und <Return>.

Erscheint eine Fehlermeldung beim Zugriff auf die Hilfedatei, so überprüfen Sie bitte Ihre Installation. Der Hilfetext GDS_35.HLP muß im gleichen Verzeichnis wie GDS35.EXE stehen, bei Netzwerkinstallationen sollten die Dateien das Attribut 'SHARE' besitzen.

Bitte beachten Sie auch in der unteren Statuszeile, daß diese ebenfalls kontextsensitiv gesetzt wird, um Ihnen eine zusätzliche, schnelle Bedienbarkeit per Maus zu gewähren.

In den folgenden Teilen werden nunmehr die Besonderheiten der Menüs Compiler, Simulation, Programm und Optionen erklärt, da diese in engem Zusammenhang mit GDS als Hardwarebeschreibungswerkzeug stehen.

3.1 Compilierung via GDS

Wie bereits erwähnt, bietet GDS eine bekannte Oberfläche, so daß der Aufruf des Compilermenüs mittels Maus bzw. Tastatur selbsterklärend ist.

GDS prüft den gesamten Sourcecode auf syntaktische Richtigkeit, falls Sie einen der Menüpunkte *NUR SYNTAXCHECK* oder *ASSEMBLIEREN* anwählen. Verlief diese Prüfung ohne Fehler oder Warnungen (bei Fehlerlevel 1) bzw. nur mit Warnungen (bei Fehlerlevel 0, siehe Kap. 3.4 Optionen), so wird das Pinout des GALs als 1. Ergebnis der Assemblierung in einer lindgrünen Farbe (bei entsprechend eingestellter Farbdarstellung) präsentiert; Abb. 3-1 zeigt diesen Fall, Sie als Nutzer können dies nun bestätigen (OK-Schalter) oder verwerfen (Abbruch).

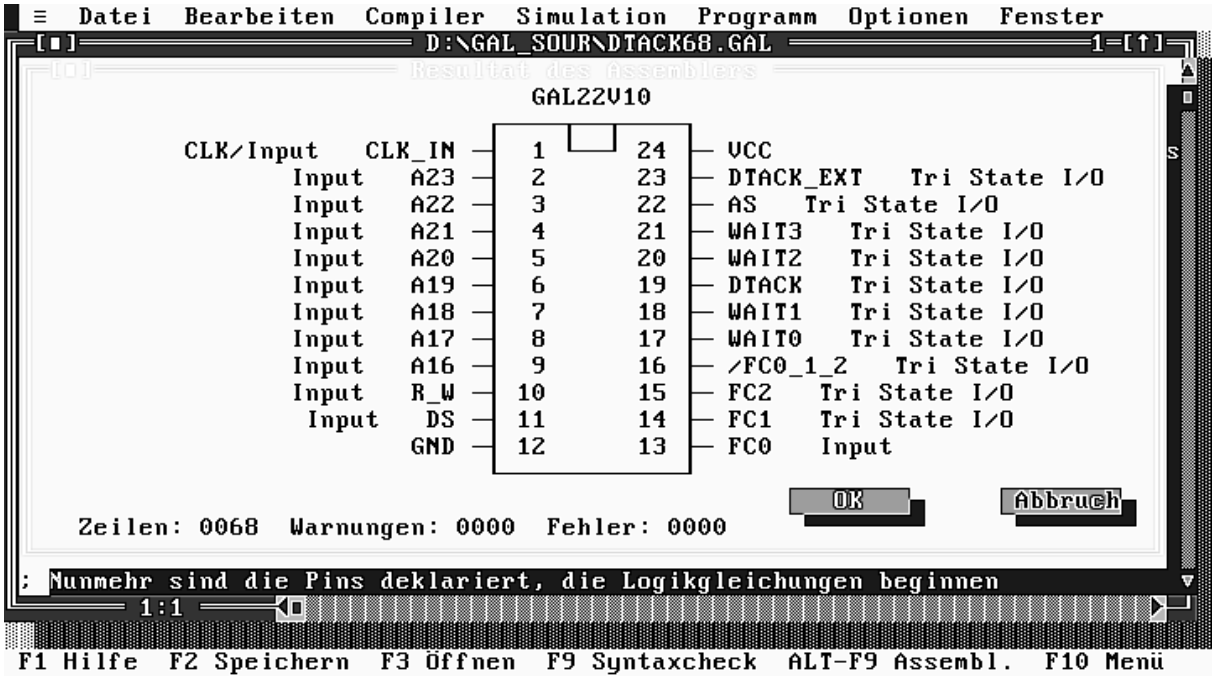

Datei Bearbeiten Compiler Simulation Programm Optionen Fenster
D:\GAL_SOUR\DTACK68.GAL
Resultat des Assemblers
GAL22V10
CLK/Input CLK_IN 1 24 VCC
Input A23 2 23 DTACK_EXT Tri State I/O
Input A22 3 22 AS Tri State I/O
Input A21 4 21 WAIT3 Tri State I/O
Input A20 5 20 WAIT2 Tri State I/O
Input A19 6 19 DTACK Tri State I/O
Input A18 7 18 WAIT1 Tri State I/O
Input A17 8 17 WAIT0 Tri State I/O
Input A16 9 16 /FC0_1_2 Tri State I/O
Input R_W 10 15 FC2 Tri State I/O
Input DS 11 14 FC1 Tri State I/O
GND 12 13 FC0 Input
OK
Abbruch
Zeilen: 0068 Warnungen: 0000 Fehler: 0000
; Nunmehr sind die Pins deklariert, die Logikgleichungen beginnen
1:1
F1 Hilfe F2 Speichern F3 Öffnen F9 Syntaxcheck ALT-F9 Assembl. F10 Menü

Abb. 3-1: Pinout bei erfolgreicher Assemblierung

Beide Aktionen weisen Kurzbefehle auf, <F9> für *NUR SYNTAXCHECK*, <ALT-F9> für *ASSEMBLIEREN*. Der Unterschied zwischen beiden Aktionen liegt in der weiteren Ausführung nach erfolgreicher Assemblierung; *NUR SYNTAXCHECK* kehrt in den Editor des soeben geprüften Sourcecodes zurück, während *ASSEMBLIEREN* als Ergebnis in einem neuen Fenster den JEDEC-Sourcecode generiert, es sei denn, im Pinout-Fenster wurde die Aktion abgebrochen.

Der erhaltene JEDEC-Code entspricht dem Standard 3A und kann für den internen Programmiereranschluß oder für beliebige externe Programmiergeräte (siehe Kap. 3.3) genutzt werden.

Eine fehlerhafte Assemblierung bricht jedoch den gesamten Vorgang ab; Sie als Nutzer können dies sofort bemerken, da die Farbe des Pinout-Fensters auf Rot wechselt und natürlich die Anzahl der Fehler sowie Warnungen anzeigt. GDS wechselt nach Ihrer Bestätigung in den speziellen Fehlereditormodus, wobei durch Abbruch dies vorzeitig verhindert werden kann.

Abb. 3-2 zeigt einen solchen Fehlerfall. Im unteren Meldungsfenster sind der Fehlertyp und die Zeilennummer des Fehlers im Sourcecode in jeweils einer Zeile aufgeführt, im oberen Fenster, der den Sourcecode enthält, wird die fehlerhafte Zeile invers unterlegt, um Ihnen die Fehlerbehebung zu erleichtern.

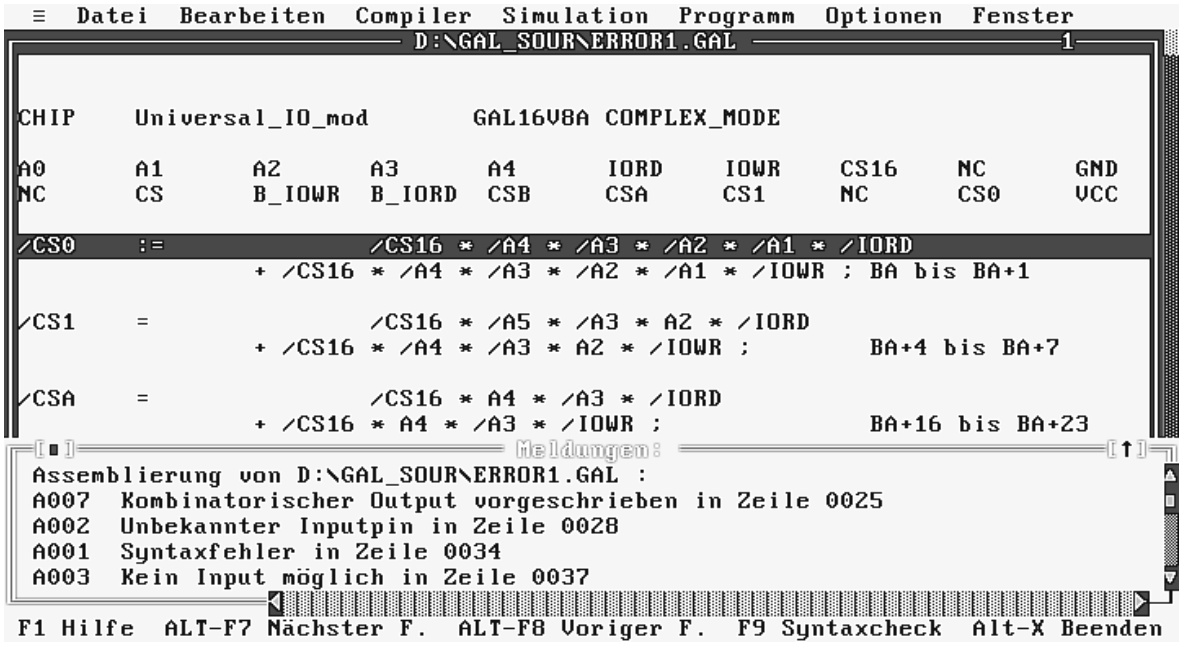

≡ Datei Bearbeiten Compiler Simulation Programm Optionen Fenster
D:\GAL_SOUR\ERROR1.GAL 1
CHIP Universal_IO_mod GAL16V8A COMPLEX_MODE
A0 A1 A2 A3 A4 IORD IOWR CS16 NC GND
NC CS B_IOWR B_IORD CSB CSA CS1 NC CS0 VCC
/CS0 := /CS16 * /A4 * /A3 * /A2 * /A1 * /IORD
+ /CS16 * /A4 * /A3 * /A2 * /A1 * /IOWR ; BA bis BA+1
/CS1 = /CS16 * /A5 * /A3 * A2 * /IORD
+ /CS16 * /A4 * /A3 * A2 * /IOWR ; BA+4 bis BA+7
/CSA = /CS16 * A4 * /A3 * /IORD
+ /CS16 * A4 * /A3 * /IOWR ; BA+16 bis BA+23
Meldungen:
Assemblierung von D:\GAL_SOUR\ERROR1.GAL :
A007 Kombinatorischer Output vorgeschrieben in Zeile 0025
A002 Unbekannter Inputpin in Zeile 0028
A001 Syntaxfehler in Zeile 0034
A003 Kein Input möglich in Zeile 0037
F1 Hilfe ALT-F7 Nächster F. ALT-F8 Voriger F. F9 Syntaxcheck Alt-X Beenden

Abb. 3-2: Fehlereditormodus

Um auf den nächsten Fehler im Sourcecode zu wechseln, gibt es mehrere Möglichkeiten:

Ist das Meldungsfenster aktiv, so können Sie mit den <Cursor Up>- und <Cursor Down>-Tasten die unterlegte Fehlermeldungszeile verändern; eine einfache Betätigung der <SPACE>-Taste wechselt im Sourcecodefenster auf die zugehörige Zeile, bei Betätigen der <RETURN>-Taste wechseln Sie direkt in den Editor.

Der Wechsel kann auch durch Anwahl der Meldungszeile mit dem Mauscursor und zweimaliges Anklicken mit der linken Maustaste erfolgen. Ferner erreichen Sie den nächsten bzw. vorhergehenden Fehler durch die Tasten <ALT-F7> und <ALT-F8>, deren Kürzel auch in der Statuszeile zur Mausbedienung (Anfahren mit dem Mauscursor und Anklicken mit der linken Taste) vorhanden sind.

Die weiteren Menüpunkte dienen der Steuerung der Assemblierung. *PRECOMPILIEREN* öffnet ein neues Fenster, in das der Assemblercode nach einem ersten Durchgang geschrieben wird. Dieser Code ist von allen Kommentaren befreit, Trennzeichen zwischen Wörtern sind als einfaches Leerzeichen ausgeführt, vor allem wird aber eine einheitliche Syntax durch Ersetzen aller Textmakros und aller Zweitfunktionszeichen für die Verknüpfungen (Einzelheiten siehe Kapitel 4) erreicht.

Der precompilierte Code ist sehr nützlich, da er zur Fehlersuche von versteckten Fehlern z.B. durch die Makroersetzung herangezogen werden kann. Selbstverständlich kann dieser Code direkt assembliert werden, da er eine Art Minimalcode darstellt.

Weiterhin können Sie einen *MINIMIERER* ausrufen. Durch diesen Aufruf werden - zunächst ohne Aufruf weiterführender Minimierungsalgorithmen - die Ersetzung aller Makros, die Entklammerung und ein ersten Syntaxcheck durchgeführt. Als Ergebnis erhalten Sie ein neues Textfenster mit dem umgewandelten Code unter der Bezeichnung <NAME>.MIN.

Dieser minimierte Code ist besonders dann wertvoll, wenn Sie Fehler in der Makroauflösung und in der Klammerung beseitigen wollen.

ABBRUCH BEI ... läßt Sie für diese GDS-Sitzung bestimmen, bei welchem Level, lediglich bei Fehlern oder auch bei Warnungen, ein Assemblierungsvorgang mit Start des Fehlereditormodes abgebrochen werden soll. Diese Einstellung wird nicht in einem Initialisierungsfile hinterlegt, wie dies bei Wahl innerhalb der OPTIONEN der Fall wäre. Einzelheiten hierzu sind in Abschnitt 3.4 aufgeführt.

16V8-MODUS bestimmt, ebenfalls für diese Sitzung, welcher Modus für die kombinatorische Einstellung der GALs 16V8 und 20V8 als grundlegend eingestellt werden soll. SIMPLE MODE dient dabei der Emulation der ersten PAL-Generation, COMPLEX MODE der der 2. Generation. Die Grundeinstellung läßt sich jederzeit im Sourcecode überschreiben, indem in der 'CHIP'-Zeile nach dem GAL-Typ der Modus angegeben wird. Zu diesem Thema sind Einzelheiten in den Kapitel 4 und 5 näher erläutert.

Das Meldungsfenster können Sie mit dem letzten Menüpunkt *FEHLEREDITOR* aufrufen. Dieses Meldungsfenster kann im übrigen nicht gelöscht, sondern nur versteckt werden, so daß die letzten eingetragenen Fehlermeldungen mitsamt der Querverweise noch enthalten sein können.

3.2 Simulationen des Source- und JEDEC-Codes

Ein syntaktisch korrekter Sourcecode bzw. der JEDEC-Code kann mit Hilfe des integrierten Simulators bereits vor dem Programmierungsvorgang simuliert werden. Die Simulationen innerhalb von GDS V3.5 beziehen sich dabei auf die logische Ebene; Laufzeitsimulationen, die also insbesondere bei zeitkritischen Applikationen die korrekte Funktion eines GALs simulieren, sind derzeit nicht in GDS integriert.

Im Menü SIMULATION finden Sie zwei Untermenüpunkte zum Starten sowie einige zur Grundeinstellung des Simulationsvorgangs. Starten Sie die Simulation erstmalig, dann können Sie dies sowohl mittels *SIMULATION* als auch mit *EINZELPINAUSWAHL* tätigen. GDS präsentiert Ihnen einen Auswahlbildschirm, in dem Sie alle Ausgangspins oder einen Einzelpin mit seinen gesamten Abhängigkeiten mittels Tastatur oder Maus wählen können. Abb.3-3 zeigt diesen Fall für eine Sourcecodesimulation, bei der die Ausgangspins natürlich mit dem von Ihnen gewählten Namen versehen sind:

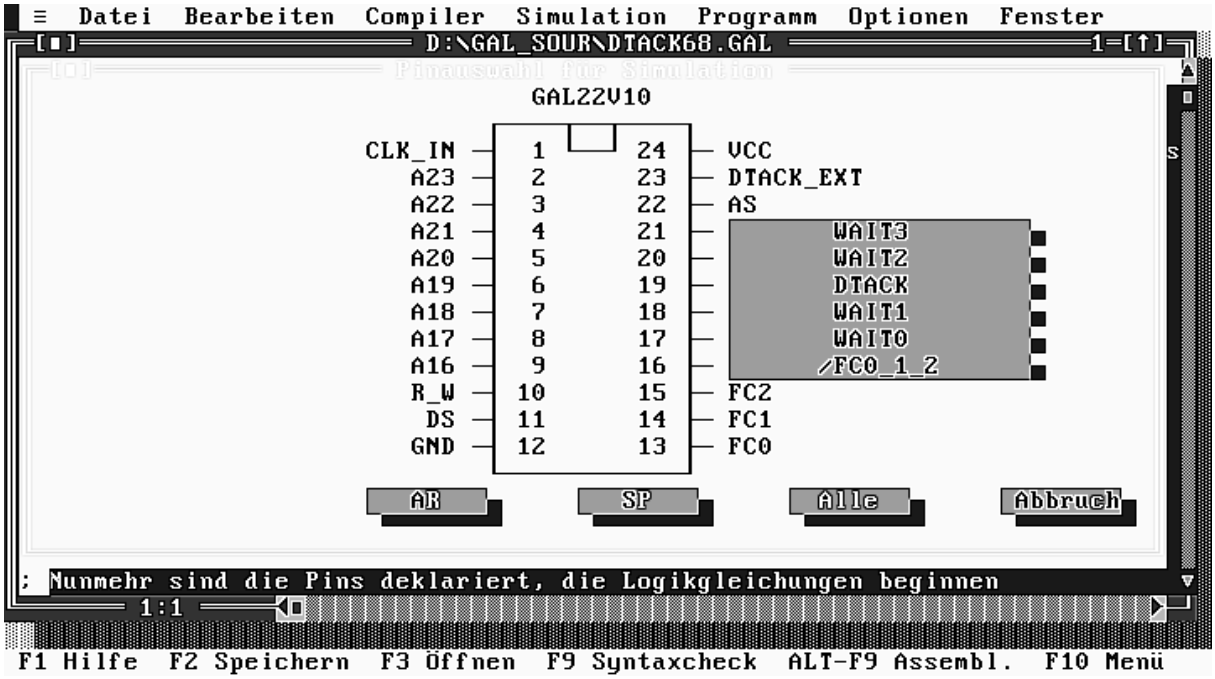


Abb. 3-3: Pinauswahlbildschirm

Die nun startende Simulation läuft komplett in einem speziellen Fenster ab, in das Sie Vorgaben zu den Belegungen per Tastatur oder Maus eingeben können, und in dem die Simulationsergebnisse entsprechend präsentiert werden. In Abb. 3-4 wurde ein Einzelpin zu diesem Zweck ausgewählt.

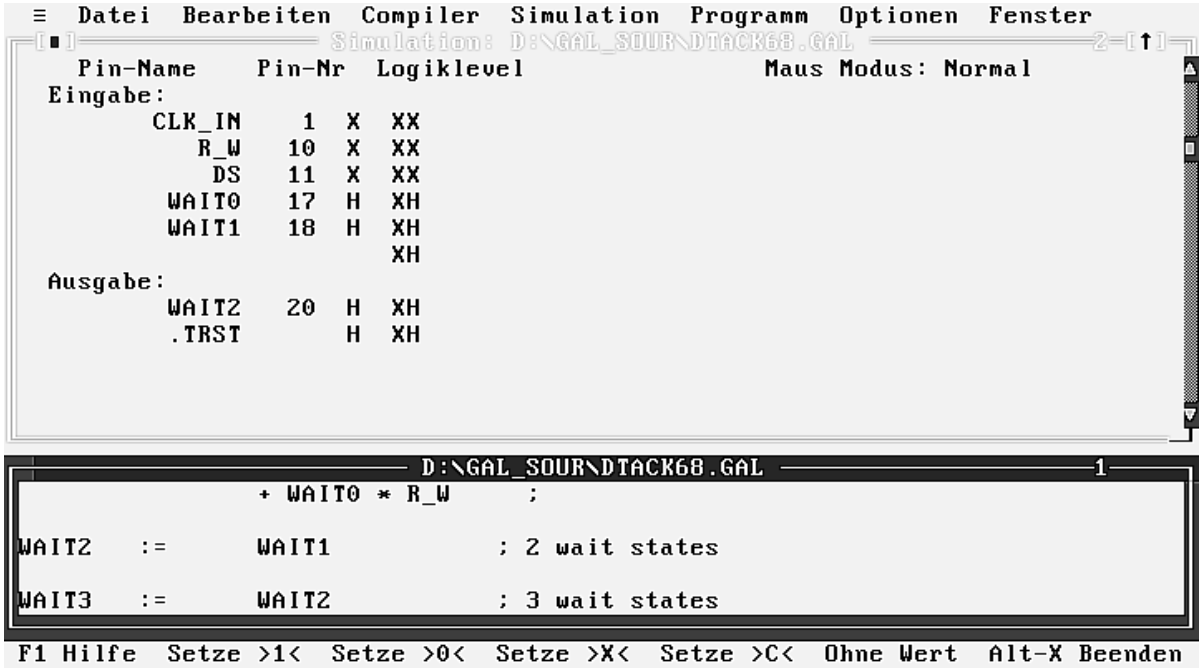

≡ Datei Bearbeiten Compiler Simulation Programm Optionen Fenster
Simulation: D:\GAL_SOUR\DTACK68.GAL
Pin-Name Pin-Nr Logiklevel Maus Modus: Normal
Eingabe:
CLK_IN 1 X XX
R_W 10 X XX
DS 11 X XX
WAIT0 17 H XH
WAIT1 18 H XH
XH
Ausgabe:
WAIT2 20 H XH
.TRST H XH
D:\GAL_SOUR\DTACK68.GAL
+ WAIT0 * R_W ;
WAIT2 := WAIT1 ; 2 wait states
WAIT3 := WAIT2 ; 3 wait states
F1 Hilfe Setze >1< Setze >0< Setze >X< Setze >C< Ohne Wert Alt-X Beenden

Abb. 3-4: Simulationsfenster bei Einzelpinwahl

Die Vorbelegung aller Eingangspins ist zunächst ein 'X', was einen offenen Eingang symbolisiert; dessen Interpretation im GAL entspricht allerdings einem Low-Eingang.

Der oder die Ausgänge werden ab der zweiten Spalte als berechnete Ergebnisse auf dem Bildschirm präsentiert. 'H' und 'L' stellen die entsprechenden High und Low-Pegel dar, während ein 'X' an einem Ausgangspin den Tristatezustand, also den undefinierten Pegel durch Abschalten der Ausgangstreiber darstellt.

Haben Sie die Einzelpinsimulation ausgewählt, dann wird der Zustand des Ausgangspins einschließlich aller möglichen Steuersignale, .TRST für den Tristatetreiber, .CLCK für die Taktsteuerung, .ARST für den asynchronen Reset, .APRST für den asynchronen Preset (letztere drei Steuersignale nur für den GAL20RA10), ausgegeben. Außerdem wird in dem entsprechend verkleinerten Editorfenster unterhalb des Simulationsfensters der zuständige Sourcecode dargestellt; in dieses Fenster können Sie jederzeit springen und Veränderungen vornehmen, beim Rücksprung in die Simulation wird nach Sourcecodeveränderung allerdings die Simulation neu gestartet.

Alle Veränderungen von Fenstergrößen etc. werden bei Schließen der Simulation rückgängig gemacht. Außerdem sollte erwähnt werden, daß maximal 4 Simulationsfenster geöffnet sein können und daß pro Sourcecodefenster nur eine Simulation gestartet werden kann.

Als Tastatureingaben sind nunmehr

'0' oder 'L'	für einen Low Pegel,
'1' oder 'H'	für einen High Pegel,
'C'	für einen normalen Takt (zunächst low, dann high),
'K'	für einen invertierten Takt und

'X' für einen offenen Eingang

zulässig. <RETURN> berechnet die nächste Simulationsspalte, wobei ggf. alle Simulationsspalten um eine nach links gescrollt sowie die erste verworfen werden. Nicht vorbelegte Simulationszellen erhalten den Wert des Vorgängers.

Jede Eingabe wird rechts von der Stelle im 'Fadenkreuz' eingefügt bzw. es wird die Stelle rechts davon überschrieben. Ändern Sie bereits simulierte Spalten, so wird die Simulation mit den neuen Eingaben wiederholt.

Mauseingaben können in analoger Weise die Simulation beeinflußen. In der Statuszeile finden Sie zu diesem Zweck mehrere Vorbelegungen des Mauscursors, dessen Zustand im Simulationsfenster rechts oben angezeigt wird, in Abb. 3-4 z.B. durch *Maus Modus: Setzt C*. Ein einfaches Anklicken mit dem Mauscursor einer Stelle im Simulationsfenster setzt dann an dieser Stelle den entsprechenden Zustand, in diesem Fall sogar zwei, low und high.

Die andere Methode, per Maus ganze Eingabezeilen zu konfigurieren, besteht in dem Anklicken einer Position mit Halten der linken Maustaste. Fahren Sie nun nach rechts oder links, so wird eine komplette Zeile mit Vorgaben belegt, und zwar mit 'H', falls sich die vertikale Position des Mauscursors oberhalb, mit 'L', falls unterhalb, und mit 'X', falls übereinstimmend mit der Startposition. Nach Loslassen der linken Maustaste wird die Zeile bis zur neuen horizontalen Position mit der Vorgaben ausgefüllt und die Simulationsspalten ggf. bis dahin berechnet.
Die nächste Simulationsspalte erhalten Sie per Mausklick im freien Simulationsraum.

Nachdem nun die Grundzüge der Simulation erklärt worden sind, folgen jetzt die weiteren Menüpunkte im Simulationsmenü.

Der Unterschied zwischen *SIMULATION* und *EINZELPINAUSWAHL* wird in dem Verhalten bei bereits begonnener Simulation deutlich. Betrachten Sie beispielsweise den JEDEC- oder Sourcecode in dem entsprechenden Fenster und wählen nun *SIMULATION* aus, so springen Sie lediglich in das bereits geöffnete Simulationsfenster, es sei denn, eine Sourcecodeänderung trat seit Simulationsbeginn auf. *EINZELPINAUSWAHL* hingegen startet in jedem Fall die Simulation neu.

Mit *AUTO-BASIERT* (Standard), *GAL-BASIERT* oder *JEDEC-BASIERT* beeinflußen Sie die Wahl des der Simulation zugrundeliegenden Codes. Im Normalfall sollte dies immer *AUTO-BASIERT* sein, da GDS dann selbst detektieren kann, ob es sich um GAL-Sourcecode oder JEDEC-Code handelt, und welcher vorgewählte IC-Typ simuliert werden soll.

In wenigen Ausnahmefällen kann GDS diese Autodetektion nicht vornehmen. In diesem Fall müssen Sie dann vorgeben, ob es sich um Sourcecode oder JEDEC-Code handelt. *AUTO-BASIERT* schaltet dann immer um eine Stufe weiter, wenn Sie diesen Punkt anwählen.

IC-TYP läßt Sie den zugrundeliegenden IC-Typ für die Simulation auswählen. Diese Wahl ist lediglich wichtig, wenn Sie JEDEC-BASIERT vorgegeben haben; ansonsten ist GDS immer in der Lage den GAL-Typ selbständig zu erkennen.

DARSTELLUNG läßt Sie zwischen der textuellen Darstellung (standardmäßige Einstellung) und einer Semigrafik umschalten. Semigrafische Darstellungen sind insbesondere bei Buszyklusdarstellungen etc. sehr nützlich.

AUTO-CLK schaltet eine automatische Generierung von Taktsignalen an Pin 1 ein (standardmäßig) bzw. aus. Diese Funktion ist beim 20RA10 nicht möglich, da dieser GAL

keinen fest zugeordneten Takteingang besitzt. Für alle anderen Bausteine ist es sehr bequem, bei getakteten GALs die Taktgenerierung an Pin 1 automatisch vornehmen zu lassen, was ansonsten per Hand einzeln nachgeführt werden müßte.

AUTO-TEST-VEKTOREN bestimmt, ob die Simulationsergebnisse gleichzeitig als Testvektorenfile mitgeschrieben werden sollen oder nicht (Standard). Der Testvektorenfile kann mit geeigneten Programmiergeräten zum direkten Test von GALs benutzt werden, indem das Programmiergerät an alle vorgesehenen Eingabepins die entsprechenden Pegel legt und die Ausgaben mit den vorberechneten (in diesem Fall simulierten) Werten vergleicht.

3.3 Programmierung eines GALs

Die Bedienung des Programmiergeräts zum Brennen des GALs ist für den GDSProg, den GDSProg2 und des hed.chip in GDS Version 3.5 fest integriert, so daß die Sequenz Editierung - Assemblierung - Simulation - Programmierung ohne jeden Wechsel der Bedienungs-umgebung durchlaufen werden kann.
Andere Programmiergeräte können mit Hilfe einer Aufrufschnittstelle in GDS eingebunden werden. Die Wahl eines Programmieranschlußports für den GDS-Prog sowie die Vorgabe der Aufrufnamen einschließlich notwendiger Parameter wird unter OPTIONEN vorgenommen.
Bei der Bedienung des GDS-Prog werden Sie als Bediener interaktiv in den Programmierungsvorgang einbezogen. Dies geschieht durch Meldungs- und Quittierungsfenster, mit denen Sie Aktionen wie Einsetzen des GALs als abgeschlossen mitteilen.

Abb. 3-5 zeigt das Menü PROGRAMM mit den Unterpunkten *IC-TYP*, *LESE GAL*, *VERIFIZIERE GAL*, *PROGRAMMIERE GAL*, *LÖSCHE GAL*, *SETZE SECURITY FUSE* und *EXTERNE PROGRAMME*. Die Bedeutung dieser Menüpunkte sollte im wesentlichen bereits aus ihrer Bezeichnung hervorgehen, so daß nur zwei Punkte näher erläutert werden müssen:

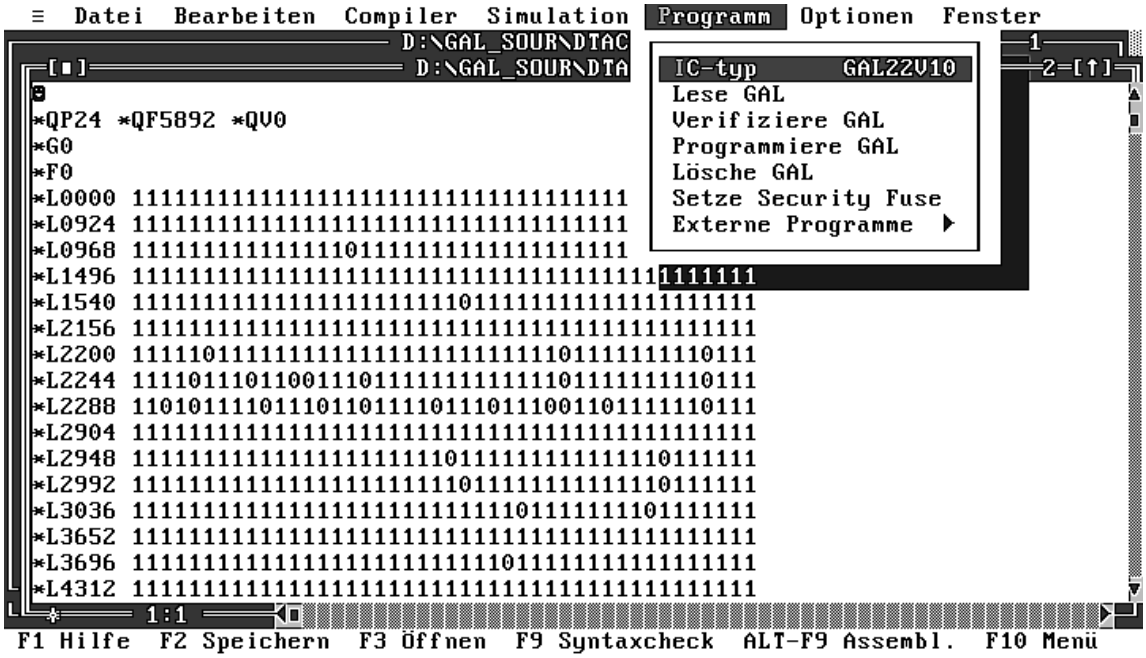


Abb. 3-5: Menü PROGRAMM

Der *IC-TYP* wird im Regelfall nicht benötigt, da GDS 3.5 auch im JEDEC-Code die benötigten Informationen erhält, um den richtigen Ziel-GAL innerhalb seines Programmiersystems auszuwählen. Diese Wahl kann im Einzelfall, wenn es sich um früher bereits assemblierten Code mit weniger internen Informationen handelt, fehlschlagen, so daß

dann - und nur dann - der korrekte IC-Typ durch Sie ausgewwählt werden muß. Sie sollten, solange Sie mit GDS 3.5 arbeiten und keinen anderweitig erzeugten JEDEC-Code programmieren wollen, diesen Menüpunkt außer Acht lassen.

SETZE SECURITY FUSE dient dem Setzen einer speziellen Sicherung, die das korrekte Funktionieren des GALs nicht beeinträchtigt, ein Auslesen aber unmöglich macht. Diese Funktion der GALs ist für Sie besonders wichtig, wenn Sie Ihre Schaltung gegen Kopieren schützen wollen. Dieser Menüpunkt bietet Ihnen die Möglichkeit, die Schutzfunktion, die übrigens auch von Ihnen nur durch Löschen des GALs wieder ausgeschaltet werden kann, jederzeit einzuschalten. Wünschen Sie dies in jedem Fall unmittelbar nach einem Programmiervorgang, etwa, weil Sie eine Serie von GALs brennen wollen, so erreichen Sie dies durch Änderung im JEDEC-Code, indem Sie eine der oberen Zeilen mit dem Inhalt '*G0' auf '*G1' ändern. Dies bedeutet für jeden Programmer, daß die Auslesesperre unmittelbar nach Programmieren und Verifizieren gesetzt werden soll.

Abb. 3-6 zeigt Ihnen als Beispiel das Auslesen eines GALs mit einem entsprechenden Meldungsfenster, in dem die Kennung des GALs ausgewiesen wird. Möchten Sie die ausgelesenen Daten in einen JEDEC-Sourcecode umwandeln, dann müssen Sie lediglich den OK-Schalter betätigen, ansonsten wird der Vorgang abgebrochen.

≡ Datei Bearbeiten Compiler Simulation Programm Optionen Fenster
Auslesen des GAL
GAL20V8 einsetzen,
GAL wird gelesen ...
Hersteller: A1
IC-Typ: 3A, Algor.: 02
OK
Abbruch
F1 Hilfe F2 Speichern F3 Öffnen F9 Syntaxcheck ALT-F9 Assembl. F10 Menü

Abb. 3-6: Lesen eines GALs

GDS prüft bei jedem Vorgang seine Kenntnis über den eingesetzten IC und unterbricht Programmier- und Löschvorgänge sofort, falls unbekannte ICs eingesetzt werden.
Mit GDS-Prog programmierbare ICs sind im Anhang D aufgeführt.

EXTERNE PROGRAMME präsentiert Ihnen ein weiteres, variables Menü, in dem Sie Ihre unter *OPTIONEN* eingestellten Programmaufrufe wiederfinden. Die Wahl eines dieser Menüpunkte bewirkt das Starten des darin benannten Programms, nach Beendigung kehrt das Programm dann an diese Stelle zurück.

3.4 Die Einstellung im Menü Optionen

Ein gewisses Maß an individueller Konfiguration kann im Menü OPTIONEN erreicht werden. Zu dieser Konfiguration gehören eine ausblendbare Uhr, die Wahl der Bildschirmfarben, die Wahl des Programmiergeräts sowie ggf. des Ports für den GDS-Prog und diverse Grundeinstellungen für die GALs. Nach Beenden der GDS-Sitzung wird die eventuell veränderte Konfiguration in einem Initialisierungsfile in dem Verzeichnis gespeichert, aus dem Sie GDS gestartet haben.

Die *Uhr* kann einfach ein- bzw. ausgeschaltet werden, indem Sie den Menüpunkt anwählen (Toggle-Funktion).

Für die *Farbwahl* erhalten Sie ein Auswahlmenü mit den Möglichkeiten *Farbig, LCD-Schirm und Monochrom*. Sie sollten eine dieser Kombinationen den Gegebenheiten Ihres Bildschirms angepaßt auswählen.
Unter *IC-Typ* wählen Sie den grundsätzlichen IC-Typ aus, der insbesondere für den Programmer als Startwert eingestellt sein soll. Diese Wahl ist für die Sequenz Editieren - Simulieren - Assemblieren - Programmieren nicht von Bedeutung, da auf die Eintragungen im Sourcecode zurückgegriffen wird.

Anders verhält es sich, wenn Sie beispielsweise einen GAL, den Sie im Programmer eingesetzt haben, auslesen wollen. Hier ist bei Benutzung des GDS-Prog die Vorwahl des GALs unerläßlich, und eben dieser Grundtyp wird an dieser Stelle eingestellt.

Sie erhalten folgendes Auswahlmenü nach Betätigung dieses Punktes:

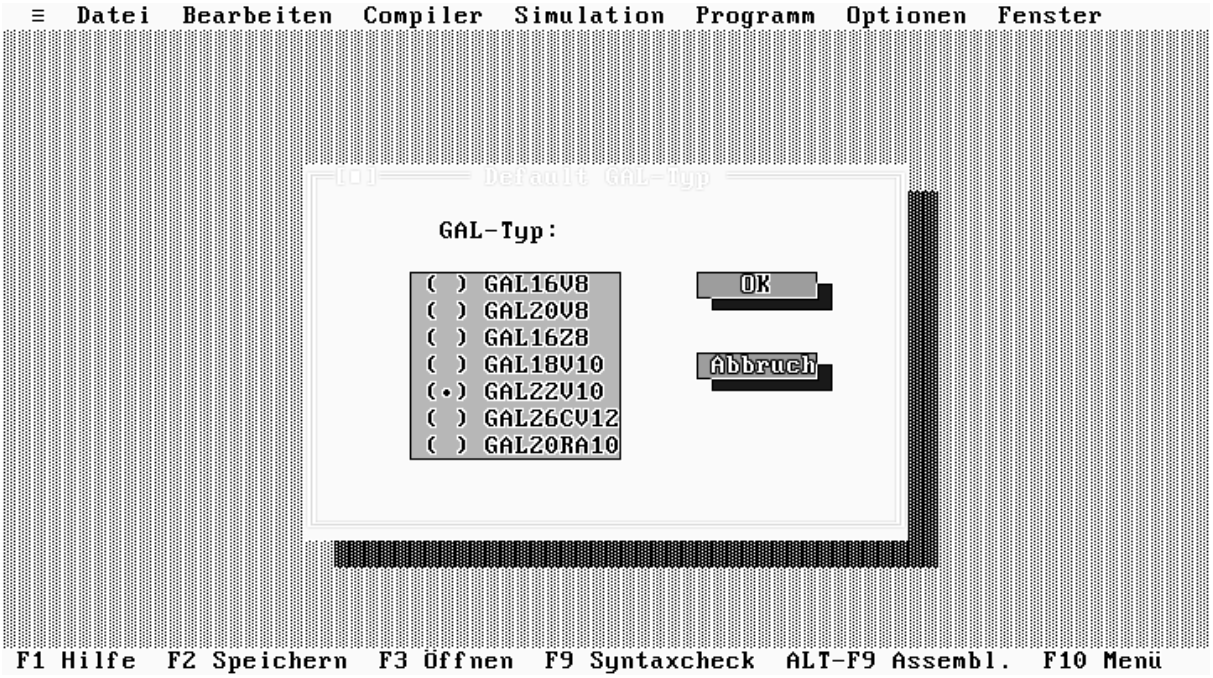


Abb. 3-7: Auswahl GAL-Typ

Wählen Sie nun den gewünschten GAL-Typ, der nach Starten von GDS V3.5 eingestellt sein soll, wobei Unterscheidungen zwischen GAL16V8 und GAL16V8A/B/D o.ä. entfallen. Ein

ähnliches Menü erhalten Sie auch unter *PROGRAMM / IC-Typ*, nur sind Eintragungen dort nach Beenden von GDS verloren, während die unter *OPTIONEN* eingetragenen Werte in dem Initialisierungsfile gespeichert werden.

Die *Port-Adresse* und das Programmiergerät wird in einem speziellen Fenster eingestellt, in dem Sie, siehe Abb. 3-8, zwischen LPT1 bis LPT3, COM1 bis COM4, einer Spezialadresse oder keinem Anschluß wählen.

Abb. 3-8 zeigt die Einstellung des Programmiergeräts und des zugehörigen Anschlusses in GDS V3.5. Zusätzlich zur Unterstützung des GDSProg ,GDSProg2 sowie des hed.chip die beide über LPTx (oder COMx mit Hilfe eines speziellen Adapters) anschließbar sind, ist auch ein Evaluationsboardanschluß an LPTx integriert.

Die EvalBoards müssen zur Lattice-Spezifikation kompatibel sein und sind dann mit Hilfe spezieller Kabel an GDS V3.5 für die Bausteine ispGAL22V10, GDS14, GDS18 und GDS22 anschließbar. Kabel sowie EvalBoards sind von SH-Elektronik lieferbar.

Die Einstellung des Anschlusses in GDS muß sowohl für den Programmierer wie für den Anschluß geschehen, wobei bei ungültigen Kombinationen automatisch *'Kein Anschluß'* gewählt wird. Im Pull-Down-Menü selbst wird der Port angegeben, der eingestellte Programmierer kann nur durch das in Abb. 3-8 gezeigte Bild erfahren werden.

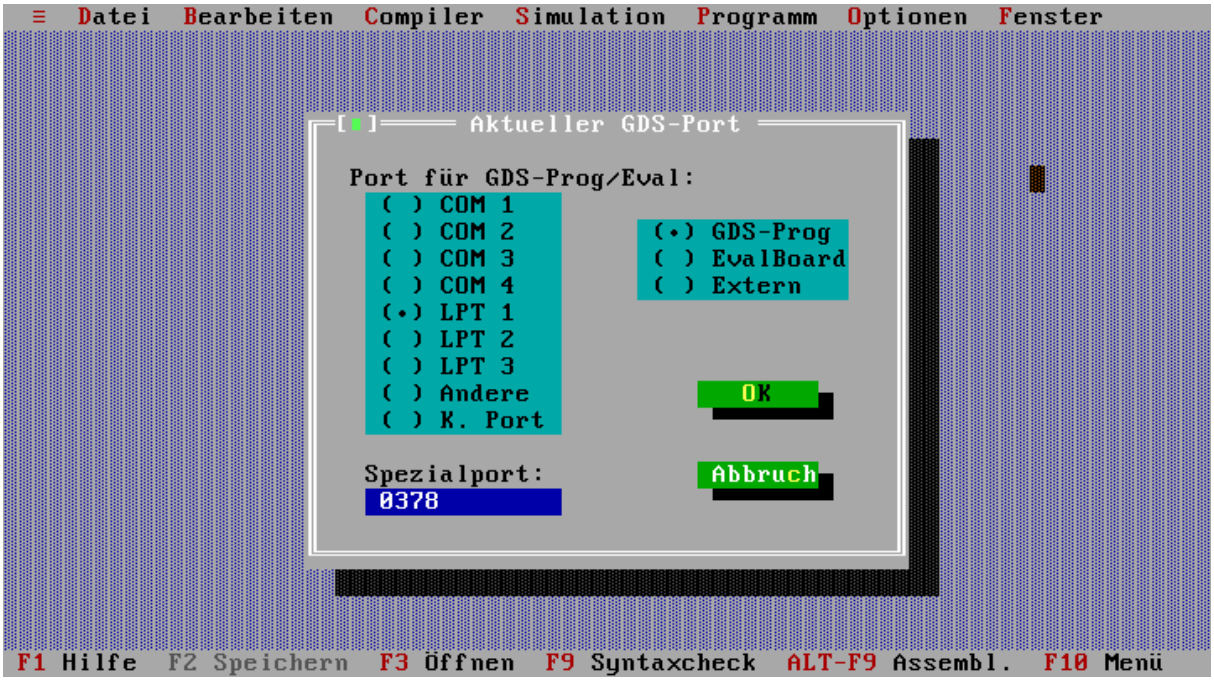


Abb. 3-8: Auswahl Portadresse

LPT1 bis LPT3 sowie COM1 bis COM4 können nur angewählt werden, wenn diese als Ports im BIOS eingetragen sind, ansonsten wird keine Veränderung vorgenommen. Die Einstellung an einen Spezialport dient der Nutzbarmachung von LPT-Anschlüssen, die nicht im BIOS bekannt sind, aber eine vollständige Kompatibilität zum LPT-Standard besitzen. Sie sollten diese Adresse mit Sorgfalt einstellen, da seitens GDS nichts überprüft werden kann, somit also eine Einstellung auf eine nahezu beliebige Stelle mit unübersehbaren Gefahren vorliegen kann.

Die Anwahl von *Externe Programme* gibt Ihnen Gelegenheit zur Eingabe von bis zu 5 Programmen, die dann aus dem *PROGRAMM*-Menü heraus aufgerufen werden können. Sie können Leertasten, Parameter usw. eintragen, alle Eingaben werden im INI-File zwischengespeichert.

Mit *Abbruch bei ...* bestimmen Sie den Fehlerlevel, bei dem die Assemblierung zwar weitergeführt, schließlich aber nicht zur Erzeugung des JEDEC-Codes führt, sondern in den Fehlereditormodus springt. Eine 0 bedeutet hier, daß Fehler und Warnungsmeldungen zum Abbruch führen, eine 1 zeigt dies lediglich bei Fehlermeldungen an.

Für die GAL16V8-Familie muß zusätzlich der kombinatorische Modus bestimmt werden, in dem der GAL betrieben werden soll. Im Sourcecode kann dies durch die optionalen Schlüsselwörter SIMPLE_MODE und COMPLEX_MODE geschehen, wie in Kapitel 4 näher ausgeführt und in Kapitel 5 in der inneren Struktur der GALs erklärt wird.

Die grundsätzliche Einstellung eines *Modus 16V8* kann an dieser Stelle im *OPTIONEN*-Menü vorgenommen werden. Dies hat zur Folge, daß alle Sourcefiles, die die GAL16V8-Familie mit den Mitgliedern 16V8, 20V8 und 16Z8 betreffen und ohne SIMPLE_MODE bzw. COMPLEX_MODE in der GAL-Definitionszeile formuliert sind, in dem eingestellten Modus assembliert werden.

Die Einstellung im *OPTIONEN*-Menü wird wiederum im Initialisierungsfile gespeichert, während Einstellungen im *COMPILER*-Menü nur bis zur Beendigung von GDS ihre Gültigkeit behalten und dann verworfen werden.

Speichern Optionen schließlich speichert unmittelbar die eingestellten Optionen in dem momentanen Verzeichnis. Auf diese Weise können Sie durchaus verschiedene Konfigurationen in unterschiedlichen Verzeichnissen speichern.

Einführung in die GAL-Assembler Syntax

Während Kapitel 3 einer Einführung in die Bedienung von GDS, speziell der Teile COMPILER, SIMULATION, PROGRAMMIERUNG und OPTIONEN, gewidmet war, folgt nunmehr eine Einführung in die GAL-Assembler Syntax GALASM, die Hardware-Beschreibungssprache der Schaltung für GDS.

GALASM lehnt sich an PALASM2 an; so werden z.B. folgende Keywords benutzt:

CHIP	Beginn des Sourcecodes
SIMPLE_MODE	Simple Mode für GAL16V8-Famlie
COMPLEX_MODE	Complex Mode für GAL16V8-Famlie
GND	Groundpin des GALs
VCC	+5V-Pin des GALs
/OE	Output-enable Pin
CLK	Spezieller Clockinput
NC	Pin ist nicht belegt
.TRST	Endung für Tristate Definition
.CLCK	Endung für Clock Definition
.ARST	Endung für Asynch. Reset Definition
.APRST	Endung für Asynch. Preset Definition
MODE	Spezialpins für GAL16Z8, die der
SDIN	In-System-Programming des GALs
SDOUT	dienen
SCLK	
AR	Asynchronous Reset (intern für 18V10, 22V10 und 26CV12)
SP	Synchronous Preset (wie AR)
=	Kombinatorische Outputdefinition
:=	Register-Outputdefinition
+ bzw. |	'or'-link
* bzw. &	'and'-link
/ bzw. !	Negation eines Pegels
#DEFINE	Definition eines Textmakros
USER_ID =	Definition einer Kennung

Tabelle 4-1: Keywords der GALASM Syntax

Bitte beachten Sie, daß die Keywords in Großbuchstaben geschrieben werden müssen (Ausnahme: #DEFINE).

Die GALASM-Syntax folgt generell folgenden Regeln:

1. Zu Beginn des Sourcecodes können Sie einen beliebigen Text eingeben, solange Sie das Keyword 'CHIP' (in Großbuchstaben) zu Beginn einer Zeile (!) fortlassen. Dieser Textbereich

sollte in jedem Fall zur Erklärung des folgenden Sourcecodes und zur generellen Kommentierung genutzt werden.

2. Das Keyword 'CHIP' zum Zeilenbeginn deklariert den Beginn des eigentlichen Sourcecodes. Zwingend muß danach ein Wort als freier Name für den GAL folgen, anschließend ein zweites zur Deklaration des GAL-Typs. Die einzelnen Wörter können durch <SPACE> oder <TAB> getrennt werden. Tabelle 4-2 zeigt die für GDS Version 3.5 erlaubten GAL-Typen.

GAL16V8
GAL20V8
GAL16Z8
GAL18V10
GAL22V10
GAL26CV12
GAL20RA10
PALCE16V8
PALCE22V10
ispGAL22V10
GDS14
GDS18
GDS22

Tabelle 4-2: Erlaubte PLD-Typen

Hinweis:
Bitte beachten Sie, daß zwischen GAL16V8 und 16V8A,B,D, AS usw. sowie GAL20V8 und 20V8A,B,D,AS usw. keine Unterschiede in der internen Struktur bestehen, sofern dies für die Erstellung des Sourcecodes wichtig wäre. Die Unterschiede liegen im Bereich der Programmierung, dies wird jedoch durch den integrierten Programmierteil im IC ausgelesen und automatisch eingestellt.

Optional kann nach der Deklaration des GAL-Typs der asynchrone Mode für Mitglieder der GAL16V8-Familie einschließlich des PALC16V8 durch die Schlüsselwörter SIMPLE_MODE oder COMPLEX_ MODE angegeben werden. Diese Angabe hat für die übrigen GALs keine Relevanz.

3. Nach der Deklaration des GAL-Typs erwartet der Assembler die Liste der Pinnamen (getrennt durch <SPACE>, <TAB> oder <KOMMA>) für diesen GAL. Die Länge dieser Liste muß natürlich in Übereinstimmung mit der Anzahl der Pins stehen, GND und VCC werden auf die korrekte Stellung geprüft.

Die Wahl der Pinnamen liegt bei dem Nutzer, wobei die ersten 19 Zeichen charakteristisch für den Pin sind. Selbstverständlich müssen alle Namen unterschiedlich sein mit der Ausnahme 'NC', der lediglich ein Platzhalter für 'Not Connected' darstellt.

Die Keywords sind ebenfalls von der Benutzung als normale Pinnamen ausgeschlossen. VCC kann allerdings als Synonym für die Bedingungen 'immer wahr', GND für 'immer falsch' als Pseudo-Inputpins eingesetzt werden.

Weiterhin werden die internen Namen AR und SP für die Mitglieder der GAL22V10-Familie (18V10, 22V10 und 26CV12) automatisch durch GDS deklariert, so daß eine manuelle Deklaration nicht erfolgen darf. In den Logikgleichungen dürfen Sie diese Pinnamen ohne Probleme benutzen.

4. Nach den Deklarationen für GAL-Typ und Pinnamen folgen die Logikgleichungen, die immer den Aufbauregeln

<OUTPUTPIN> = funk(<INPUTPINs>) [; <KOMMENTAR>]

oder

<OUTPUTPIN> := funk(<INPUTPINs>) [; <KOMMENTAR>]

folgen.

Semikolon und <KOMMENTAR> sind hierbei optional. <OUTPUTPIN> muß ein Pin aus der Pinliste sein, wobei natürlich an der entsprechenden Stelle der gewählte GAL-Typ ebenfalls einen Ausgang zuläßt, während nahezu alle Pins als Inputs zugelassen sind (Ausnahmen natürlich GND, VCC, NC, /OE, /PL und CLK (für GAL/PALCE 16V8/16V8A, 20V8/ 20V8A, 16Z8 und 20RA10), sowie die GAL16Z8/ispGAL22V10-Spezialpins MODE, SDIN, SDOUT und SCLK). Nicht als Eingang erlaubte Pins werden vom Assembler entdeckt und beanstandet.

funk(<INPUTPINs>) stellt eine Kombination der Inputpins mit dem Invert-Zeichen (/ oder !), dem AND-Zeichen (* oder &) und dem OR-Zeichen (+ oder |) dar. Die Reihenfolge der Prioritäten ist '/' vor '*' und '+' (bzw. der jeweiligen anderen Syntaxform), die Anzahl der möglichen OR-Verknüpfungen ist durch die jeweilige GAL-Architektur beschränkt, so daß ein Überschreiten ebenfalls eine Fehlermeldung generiert. Weiterhin ist eine Klammerung erlaubt, mit der Sie die Prioritätenreihenfolge der Operatoren überschreiben können.

Die ***Klammerung*** von Ausdrücken innerhalb der Funktionszuweisung wird mit Hilfe von runden Klammern '(' und ')' vollzogen, wobei die Anzahl der Klammern jeweils übereinstimmen muß. Prinzipiell können beliebig viele Klammern gesetzt werden (die Limitation liegt im Speicherbedarf der Auswertungsroutine), wobei die üblichen Regeln der Algebra bei der Entklammerung zum Einsatz kommen.

Die Operatoren haben bei einer Entklammerung folgende Prioritätenreihenfolge:

- Die (unäre) Invertierung (eines Eingangs oder rückgekoppelten Ausgangs) steht an oberster Stelle
- Die UND-Verknüpfung hat die zweite Priorität
- Die ODER-Verknüpfung nimmt die dritte Priorität ein
- Die abschließende Invertierung des Ausgangs steht an unterster Stelle

Die Reihenfolge ergibt sich aus den Gegebenheiten der programmierbaren Logik, da der Funktionsausdruck als Kanonisch-Disjunktive-Normalform oder Sum-Of-Products zusammengesetzt wird: Erst Invertierung, dann UND-Verknüpfung, zuletzt ODER-Verknüpfung, das Ergebnis wird dann ggf. komplett invertiert und erscheint am zugewiesenen Ausgangspin.

Diese Form der Prioritätenreihenfolge ergibt dann letztlich die bekannte Regel 'Punkt- vor Strichrechnung', falls das UND mit dem '*', das ODER mit dem '+' dargestellt wird.
Die Benutzung von '=' oder ':=' definiert den jeweiligen Outputpin als kombinatorisch (oder asynchron) bzw. registered (oder synchron); synchron bedeutet hier die Synchronisation der Outputleveländerung auf die nächste positive Flanke an dem zuständigen CLOCK-Pin (Spezifikation siehe Kap. 5). Diese Benutzung ist bei den GALs/PALCEs 18V10, 22V10, isp22V10, 26CV12 und 20RA10 freigestellt, bei 16V8, 20V8, und 16Z8 muß für ':=' der GAL als synchron durch die Deklarationen CLK an Pin 1 und /OE an Pin 11/13 definiert sein. Nur eine dieser Deklarationen generiert eine Warnungsmeldung , der GAL bleibt im asynchronen Mode. Im synchronen Mode sind '=' und ':=' erlaubt.

Die Benutzung der Invertierungszeichen '/' bzw. '!' muß sorgfältig geschehen, da dies sowohl in der Pinliste als auch in den Logikgleichungen erlaubt ist. Die Deklaration eines Pins mit dem Slash und die anschließende Benutzung als Input- oder Outputpin ohne Slash hat dieselbe Bedeutung als umgekehrt: Die Einkopplung in die Sicherungsmatrix bzw. der Outputlevel ist im Vergleich zum Inputlevel invertiert. Interessant wird nur die Frage, wie ein solcher Output wieder in die Matrix rückgekoppelt ist. Hier ist Aufmerksamkeit geboten:

Ist die Deklaration des Outputpins mit Slash, die Definition ohne, dann ist der Outputlevel gegenüber dem Resultat aus den logischen Gleichungen invertiert. Bei Benutzung dieses Pins als Input mit Slash (durch Rückkopplung in die Matrix) wird dieser Input nicht invertiert, da Deklaration und Benutzung übereinstimmen, und der Input hat denselben physikalischen Level als der Output.

Im umgekehrten Fall, also bei Deklaration ohne und Definition mit Slash, entsteht für den Outputlevel kein Unterschied, aber als Input mit Slash wird der Level nunmehr invertiert, da sich im Vergleich Deklaration und Nutzung unterscheiden.

Als Ergänzung zu den Definitionen der Outputpins muß für die asynchronen Outputs der GAL/PALCE16V8-Familie (nur im COMPLEX MODE) sowie für alle der GAL/PALCE 22V10-Familie sowie den GAL20RA10 genau eine Tristatebedingung angegeben werden, die festlegt, unter welchen Bedingungen der physikalische Level an dem Pin auch angenommen wird. In den meisten Fällen wird dies implizit immer der Fall sein, die Ausgangstreiber also nie abgeschaltet werden. GDS generiert dann dies automatisch als immer wahr, wenn keine explizite Tristategleichung angegeben wird.

Diese immer zutreffende Beschaltung kann auch durch die Bedingung <pin>.TRST = VCC erreicht werden, während <pin>.TRST = GND niemals zutreffen wird.

5. Textmakros können an jedem beliebigen Zeilenanfang nach der 'CHIP'-Zeile definiert werden. Zu diesem Zweck wird diese Zeile durch

#define <MakroName> <MakroDefinition> [; Kommentar]

beschrieben, wobei der MakroName maximal 19 Zeichen enthalten darf, die eigentlich MakroDefinition, die durch das Editorzeilenende bzw. den optionalen Zeilenkommentar beendet wird, maximal 255 Zeichen. Im übrigen gelten die gleichen Regeln wie für Pinnamen, Schlüsselwörter bzw. Pinnamen sind als MakroNamen verboten, dürfen aber naturgemäß in der MakroDefinition stehen.

Maximal 20 Makrodefinitionen sind pro Sourcecode erlaubt, ansonsten erfolgt eine Warnungsmeldung (siehe Anhang A). Eine MakroDefinition gilt von der Zeile ihrer Definition an bis zum Dateiende, es sei denn, sie wird redefiniert
(erneute Definition mit gleichem MakroNamen), so daß im Fortlauf des Sourcecodes ein anderer Inhalt ersetzt werden soll.

Innerhalb der Makrodefinition werden diejenigen Namen, die bereits als MakroName vereinbart sind, durch die MakroDefinition ersetzt (rekursive Ersetzung). Aus diesem Grund sollten Sie bei der MakroDefinition streng auf eine Trennung der Namen und Operatoren (auch durch Leerzeichen!) achten, damit die Ersetzung entsprechend verläuft. Ebenfalls ist eine gewisse Aufmerksamkeit für die Einsetzung notwendig, insbesondere, falls Sie bestimmte Rechenregeln im Sinn haben. Beispielsweise geschieht folgende Ersetzung:

```
#define ABC A + B + C
...
```

AUS_1 = ABC * D

wird ersetzt zu:

AUS_1 = A + B + C * D,

wobei Sie eventuell eher an

AUS_1 = A * D + B * D + C * D

gedacht haben. Durch eine geklammerte Definition

#define ABC (A + B + C)

kann dies jedoch leicht erreicht werden.

Im Sourcecode selbst wird jedes Vorkommen des MakroNamens durch die MakroDefinition ersetzt. Diese Ersetzung sowie die Ausblendung aller Kommentierungen kann im precompilierten Code betrachtet werden.

6. Kommentare können in GDS 3.5 durch ein Semikolon ';' zeilengebunden eingeleitet werden. Eine Blockkommentierung ist zu Beginn vor dem 'CHIP'-Kommando möglich, im eigentlichen Sourcecodeteil können Kommentarblöcke auch durch /* ... */ entsprechend der C-Syntax über mehrere Zeilen eingeklammert werden.

7. Eine persönliche Kennung wird über das Schlüsselwort USER_ID = < 8 Zeichen max.> innerhalb des Definitionsblockes eingeben. Diese Kennung wird in den GAL programmiert und hilft Ihnen bei einer späteren Identifizierung.

8. Bis auf eine strenge Einhaltung der Trennung innerhalb der Makrodefinitionen kann in GDS V3.5 auf eine Trennung zwischen Variablennamen und Operator durch <SPACE> oder <TAB> verzichtet werden.

Hinweis:
Zum besseren Verständnis schauen Sie sich bitte die Beispiele im ANHANG B an.

Details über GALs

GDS Version 3.5 unterstützt die Entwicklung von anwendungsspezifischen ICs auf vier GAL/PALCE-Familien: Die GAL/PALCE16V8-Familie, die GAL/PALCE22V10-Familie, die GAL20RA10-Familie und die GDS14-Familie. Auf den folgenden Seiten soll versucht werden, diese Familie und die einzelnen Typen (16V8, 20V8/, 18V10, 22V10, isp22V10, 26CV12, 20RA10, GDS14, GDS18 und GDS22) zu beschreiben. Dies geschieht u.a. durch Darstellung der Fusemaps, deren Kenntnis für den engagierten Benutzer unumgänglich ist; zwar gibt GDS bei Verletzung der Regeln Fehlermeldungen aus, jedoch ist die genaue Kenntnis zur Vermeidung bzw. Beseitigung eines Fehlers häufig sehr nützlich.

Zusätzliche Features der GALs wie z.B. die Speicherung einer elektronischen Signatur (UES) werden nun auch vom GDS unterstützt.

5.1 Die GAL16V8-Familie

Als die GAL16V8-Familie als erste auf den Markt kam, lag das Ziel in der Ersetzung bzw. Emulation einer Vielzahl von PALs, beginnend beim PAL10H8/L8 bis zum 16H2/ L2, weiterhin den PAL16V8 und die entsprechenden Register-PALs. Sie können dies leicht nachvollziehen, da man den PAL-Sourcecode bzw. die JEDEC-Files nehmen konnte und durch Hinzufügen der GAL-Architekturbits diesen PAL durch einen der beiden Typen GAL16V8 oder 20V8 emulieren konnte.

GDS allerdings verfolgt einen anderen Ansatz, denn die interne Architektur der GALs soll vollständig ausgenutzt werden. Dies führt dazu, daß Sie Ihren bisherigen PAL-Sourcecode, falls vorhanden weiterhin ausnutzen können, GDS kreiert dann den entsprechenden PAL-Ersatztyp, wobei sich natürlich auch Kombination wie PAL13H5, die nicht existieren, ergeben können. Die Zusammenstellung der Architekturbits gemäß Ihren Wünschen liegt bei GDS.

Die Abb.5-1 und 5-2 zeigen die interne Struktur der GALs 16V8 und 20V8, Abb.5-3a stellt die komplette Output Logic Macrocell (OLMC) dar.

Die beiden GALs bestehen aus acht I/O-Pins sowie 10 bzw. 14 Inputpins. Um die verschiedenen PAL-Architekturen emulieren zu können, sind folgende Architekturbits vorhanden:

- AC0 für Umschaltung zwischen Simple/Complex Mode
- SYN für Umschaltung asynchroner/synchroner Mode
- AC1(n) für spezifische OLMC-Modes
- XOR(n) für die Output-Polarität

5.1.1 Der Synchron-Mode

SYN schaltet generell zwischen asynchronem (SYN = 1) und synchronem Mode (SYN = 0 und AC0 = 1) um. Im synchronen Mode erhalten zwei Inputpins spezielle Aufgaben: Pin 1 wird zum CLK-Input, Pin 11/13 zum /OE-Input für alle Register-Outputs. Die übrigen 8 bzw. 12 Inputpins bleiben zur freien Verfügung.

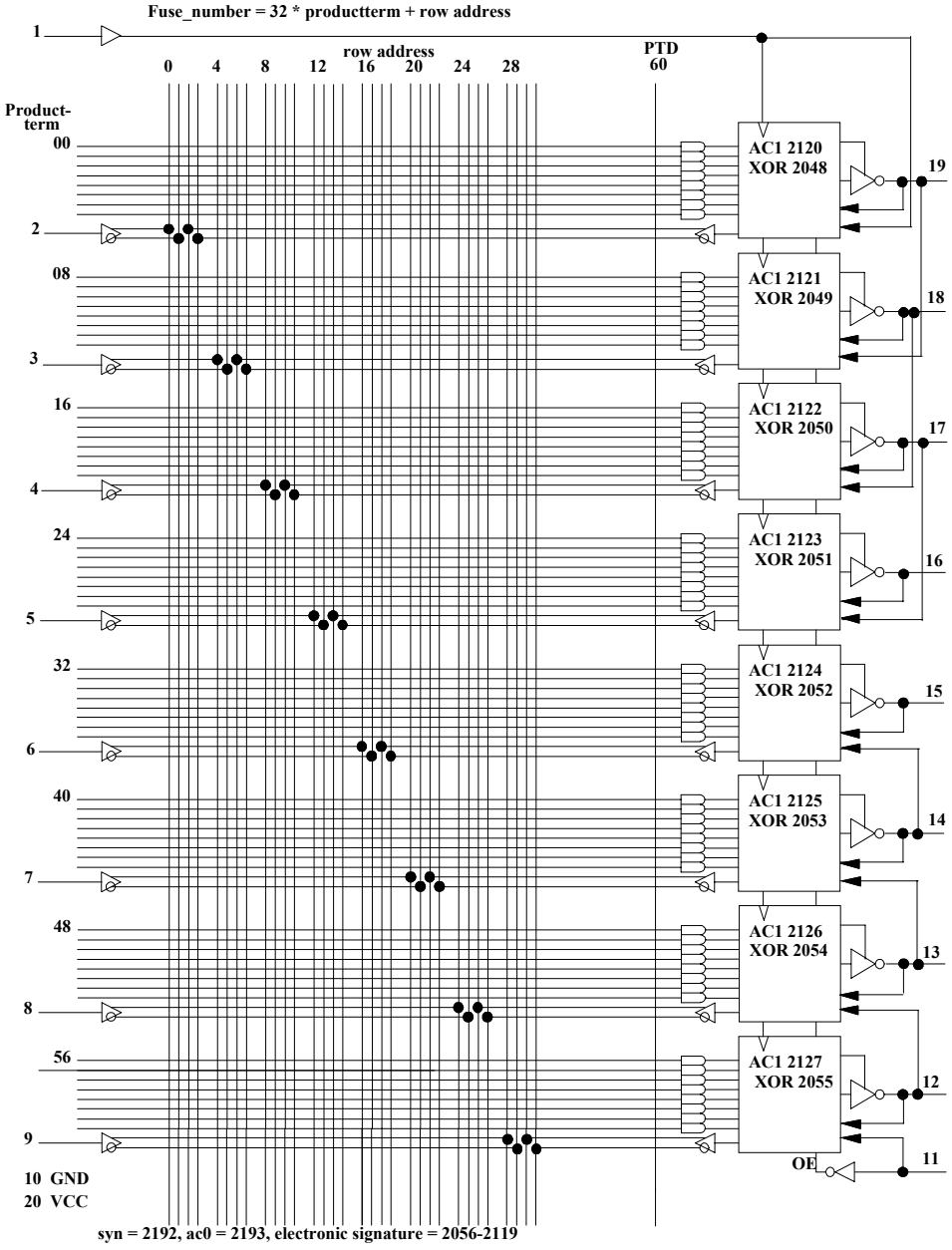


Abb. 5-1 : GAL 16V8 Fusemap

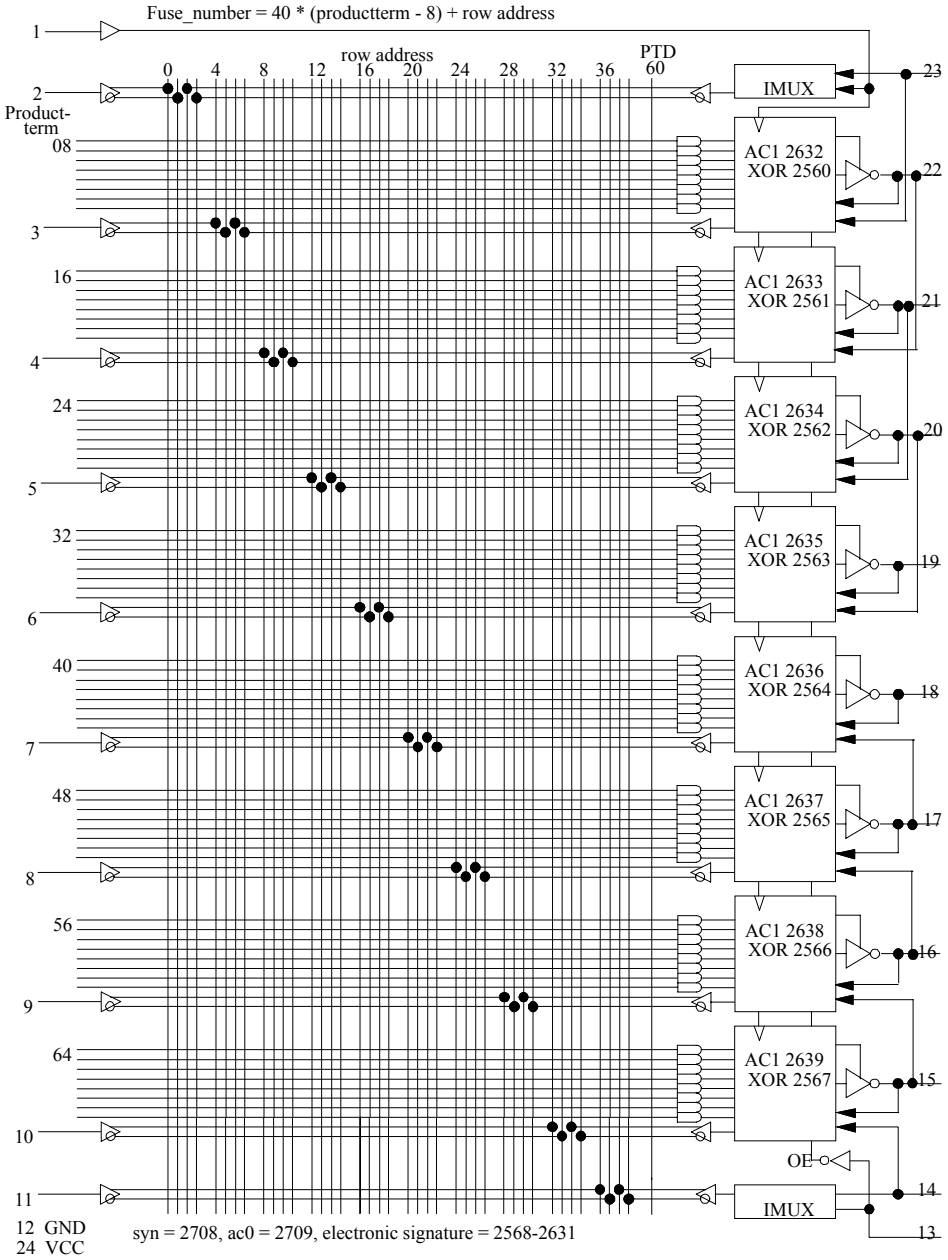


Abb.5-2: GAL 20V8 Fusemap

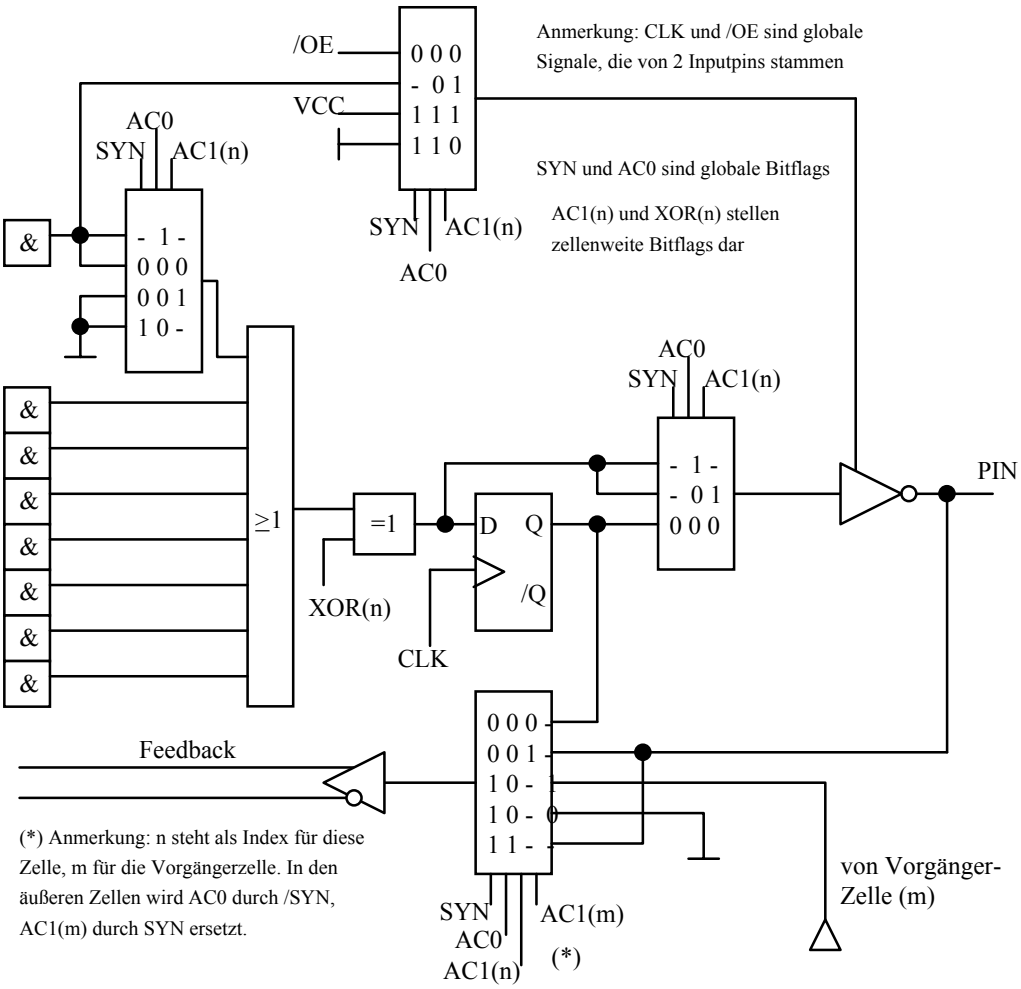


Abb.5-3a: Output Logic Macrocell OLMC des GAL/PALCE 16V8/20V8

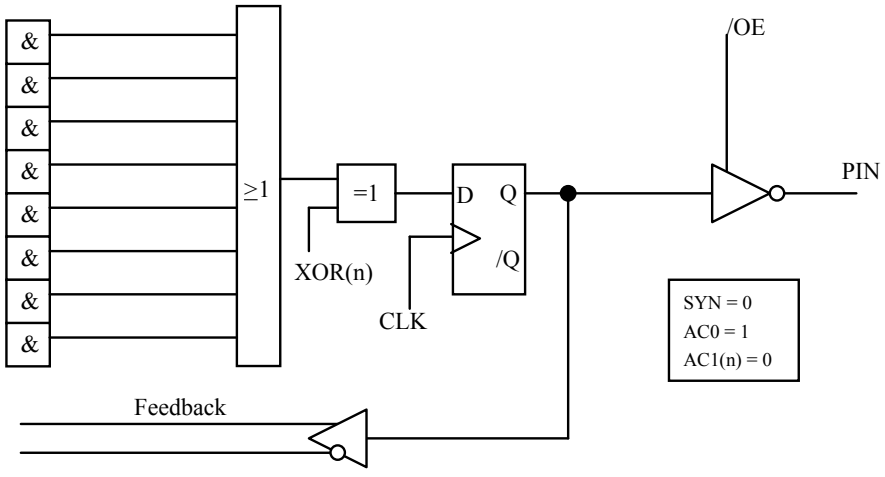


Abb.5-3b: Registered Output, aktiv high/low

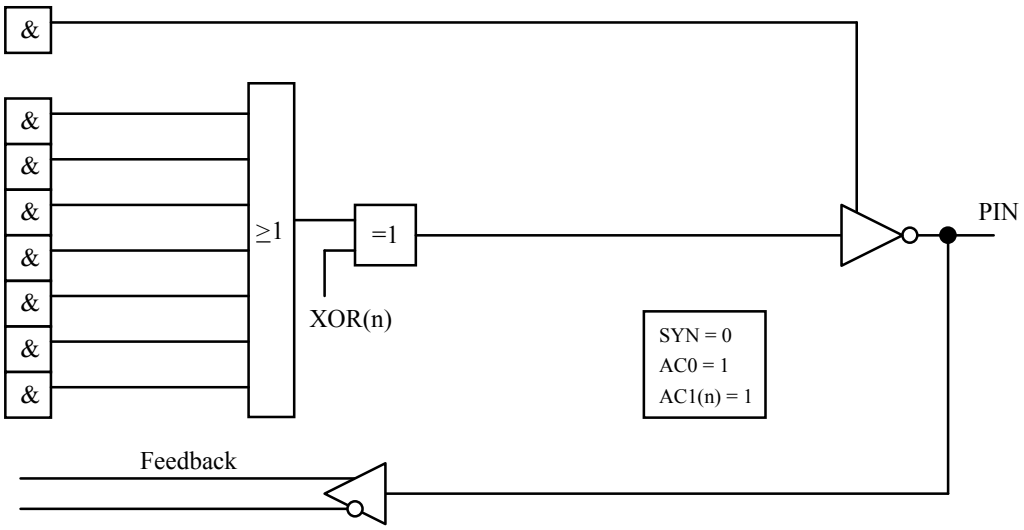


Abb.5-3c: Kombinatorischer Output in Registered GAL

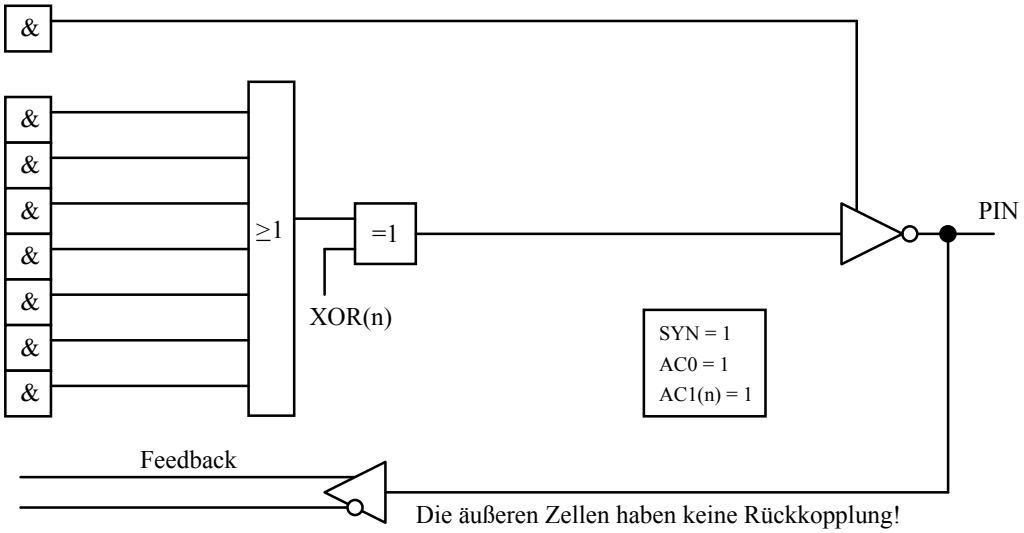


Abb.5-3d unten: Kombinatorischer Output

Die Konfiguration der I/O-Pins zeigen die Abb.5-3b und 5-3c. Sie können jeden Output als synchron definieren (SYN = 0, AC1(n) = 0), wobei dann 8 OR-Terme, aber keine dedizierte Tristate-Bedingung zur Verfügung stehen (Tristate wird durch /OE kontrolliert), oder als asynchronen Output (SYN = 0, AC1(n) = 1) mit 7 OR-Termen und spezieller Tristate Bedingung (keine Kontrolle durch /OE).

5.1.2 Der Asynchron-Mode

SYN = 1 definiert direkt den asynchronen Modus der GAL16V8-Familie, nun stehen 10 bzw. 14 dedizierte Inputpins ohne Einschränkung zur Verfügung.

Im asynchronen Mode wird zwischen dem Simple Mode (AC0 = 0) und dem Complex Mode (AC0 = 1) unterschieden. Die Ursachen hierfür liegen in dem Anspruch, eine möglichst große Anzahl von GALs emulieren zu können.

GDS unterstützt beide Modi, indem AC0 nicht automatisch von GDS gesetzt wird, sondern der Anwender dieses Bit durch Wahl des Modes in den Menüs OPTION (grundsätzliche Voreinstellung), COMPILE (gültig für die Session) oder im Sourcecode hinter der Festlegung des GAL-Typs bestimmt. Tabelle 5-1 gibt Ihnen einen Überblick über die verschiedenen Modi.

Mode	Anzahl OR-Terme	Tristate-Bed.	SYN	AC0
SIMPLE MODE	8	nein	1	0
COMPLEX MODE	7	ja	1	1
SYNCHR.MODE	7	ja für asynchr.	0	1
	8	/OE für synchron		

Tabelle 5-1: Modi der GAL/PALCE16V8-Familie

Abb.5-3d zeigt Ihnen die OLMC-Konfiguration im Complex Mode. Bitte beachten Sie, daß die äußeren Zellen (Pin 12 und 19 für den GAL 16V8/16V8A, Pin 15 und 22 für 20V8 /20V8A) keine Input-Funktion übernehmen können (einschließlich des Fehlens der Rückkopplung des Outputs). Der Versuch wird bei GDS mit einer Fehlermeldung quittiert.

Die Tristate-Bedingung kann durch eine explizite Formulierung in der Form <pin>. TRST = ..., oder implizit als immer wahr gegeben werden. Im zweiten Fall generiert GDS eine JEDEC-Zeile mit lauter '1', da dann das Resultat 'immer wahr' für die Tristate-Bedingung vorliegt.

Diese immer erfüllte Bedingung können Sie auch durch <pin>.TRST = VCC beschreiben, während <pin>.TRST = GND eine niemals erfüllte Bedingung darstellt.

Im Simple Mode erhöht sich die Zahl der OR-Terme auf 8, während kein Tristate-Signal zur Verfügung steht. Ferner sei darauf verwiesen, daß in diesem Mode die inneren OLMCs nicht rückkoppelbar sind (Pin 15/16 bzw. Pin 18/19).

Tabelle 5-2 gibt Ihnen einen Überblick der Fusenummer der GAL16V8-Familie. Bitte beachten Sie, daß die 64 Bits der User Electronic Signature (UES) frei für Ihre Nutzung sind. Im Sourcecode können Sie diese UES durch USER_ID ... beschreiben.

Tabelle 5-3 stellt summarisch die Verwendungsmöglichkeiten der GAL-Pins dar, wie sie sich aus der Architektur ergeben.

GAL/PALCE 16V8

Fuse Adresse	Funktion
0000 - 2047	AND-array
2048 - 2055	XOR-Bits für Output 19 - 12
2056 - 2119	UES zur freien Nutzung
2120 - 2127	AC1-Bits für die Outputs 19 - 12
2128 - 2191	Produktterme zur Einbindung der OR-Zeilen
2192	SYN-Bit
2193	AC0-Bit

GAL20V8

Fuse Adresse	Funktion
0000 - 2559	AND-array
2560 - 2567	XOR-Bits für Output 22 - 15
2568 - 2631	UES zur freien Nutzung
2632 - 2639	AC1-Bits für die Outputs 22 - 15
2640 - 2703	Produktterme zur Einbindung der OR-Zeilen
2704	SYN-Bit
2705	AC0-Bit

Table 5-2: JEDEC Adressen der GAL/PALCE16V8-Familie

GAL/PALCE 16V8

Pin-Nr.	*Simple Mode*	*Complex Mode*	*Registered Mode*
1	Input	Input	Clock
2-9	Input	Input	Input
11	Input	Input	/OE
12	In- oder Output	**Output**	In- und Output
13-14	In- oder Output	In- und Output	In- und Output
15-16	**Output**	In- und Output	In- und Output
17-18	In- oder Output	In- und Output	In- und Output
19	In- oder Output	**Output**	In- und Output

Bitte beachten Sie die **fettgedruckten** Ausgänge:

Anders als bei der GAL 22V10 Familie, lassen sich bestimmte Ausgänge nicht beliebig als Ein- oder Ausgänge benutzen und sind somit auch nicht rückkoppelbar. Dies ist eine häufige Fehlerquelle, bitte beachten Sie dieses bei Ihrem Design

GAL 20V8:

Pin-Nr.	Simple Mode	Complex Mode	Registered Mode
1	Input	Input	Clock
2-11	Input	Input	Input
13	Input	Input	/OE
14	Input	Input	Input
15	In- oder Output	**Output**	In- und Output
16-17	In- oder Output	In- und Output	In- und Output
18-19	**Output**	In- und Output	In- und Output
20-21	In- oder Output	In- und Output	In- und Output
22	In- oder Output	**Output**	In- und Output
23	Input	Input	Input

Table 5-3: Pinverwendbarkeit der GAL16V8-Familie

Bitte beachten Sie auch hier die **fettgedruckten** Ausgänge:

Der PALCE16V8 entspricht im logischen Aufbau und den JEDEC-Adressen nahezu 100prozentig dem GAL16V8; die komplett unterschiedliche Programmierung wird durch GDS3.5 im Programmenü abgefangen und spielt für den Einsatz der PALCEs keine Rolle.

Der einzige Unterschied wird im Simple Mode sichtbar: Die Ausgänge des PALCE16V8 sind in diesem Mode immer in die Matrix rückkoppelbar. Diese Funktion ist bei GAL16V8-ICs nur in neueren Serien zu finden, ältere können dies nicht. Aus diesem Grund werden diese Rückkopplungen beim Simple Mode, den GAL16V8 betreffend, auch mit einer Warnung versehen.

5.2 Die GAL22V10-Familie

Die zweite GAL-Generation besteht aus den drei Mitgliedern der GAL/PALCE22V10-Familie, den GALs 18V10, 22V10 und 26CV12 und dem PALCE22V10. Alle vier GAL/PALCE-Typen sind in GDS V3.5 sowohl im Assembler als auch im Programmer (GDSProg) integriert.

Der Anspruch dieser Familie ist ein anderer als der der ersten Generation. Zum GAL/PALCE 22V10 existiert ein exakter Vergleichstyp, der PAL 22V10, beide enthalten die gleiche Logikstruktur. Diese zweite Generation bietet Ihnen eine erheblich erweiterte Programmierbarkeit, zudem ist die interne Struktur sehr geradlinig.

Die Abb.5-4, 5-5 und 5-6 bieten Ihnen die Fusemaps der GAL/PALCEs 18V10, 22V10 und 26CV12 im Überblick. Für jeden GAL stehen nunmehr 10 bzw. 12 I/O-Pins sowie 8, 12 bzw. 14 dedizierte Inputpins zur Verfügung. Jede OLMC kann als synchron oder asynchron definiert werden (oder, wie erwähnt, als zusätzlicher Inputpin), wobei Pin 1 zusätzlich (und nicht alternativ) die Doppelfunktion als Input- und CLK-Pin (für die synchronen Outputs) beherbergt. Ferner kennen diese GAL-Typen keinen ausgezeichneten /OE-Input, Tristate-Bedingungen sind für jeden Output gesondert (oder implizit als 'immer wahr') zu formulieren.

Die Anzahl der OR-Terme pro OLMC variiert von 8 bis 10, 12 oder 16, je nach GAL-Typ. Dies gibt Ihnen einerseits mehr Flexibilität an die Hand, die Sie für Ihre Chip-internen Schaltungen verwenden können, andererseits kann auch ein einfacher Pinwechsel für Sie Vorteile bringen, falls die Anzahl der OR-Terme an einem Pin nicht ausreicht und GDS dies als Fehlermeldung ausgibt. Die Abb.5-7a und b zeigen Ihnen Details zu den OLMCs der

GAL/PALCE 22V10-Familie. Das Architekturbit S0 entspricht im übrigen XOR, S1 den AC1-Bits der GAL/PALCE 16V8-Familie.

Als neues Feature gegenüber der ersten Generation kennen die GALs der 22V10-Familie zwei interne 'Pins', Asynchronous Reset (AR) und Synchronous Preset (SP). GDS deklariert diese Pins automatisch, so daß Sie sie in den Logikgleichungen definieren können. AR setzt die internen Flipflops jederzeit bei Vorliegen der Verknüpfung 'wahr' zurück, so daß an den Pins der Wert high anliegt (durch den nachgeschalteten Inverter), SP setzt synchronisiert mit CLK (Pin 1) den Ausgang Q auf 1, den physikalischen Pin also auf low.

Der PALCE22V10 ist - die Logik betreffend - 100% kompatibel zum GAL22V10. Ein einziger Unterschied ist in dem Speicherplatz für eine User-ID zu detektieren; während beim GAL22V10 8 Bytes (64 Bits) als Kennung speicherbar sind, ist dies beim PALCE22V10 nicht möglich.

Die Tabelle 5-3 beschließt die internen Informationen für die GAL22V10-Familie.

	Fuse Adresse	Funktion
GAL18V10		
	0000 - 3455	AND-array
	3456 - 3475	S0/S1-Bits für die Outputs 19 - 9
	3476 - 3539	UES für freie Benutzung
GAL/PALCE22V10		
	0000 - 5807	AND-array
	5808 - 5827	S0/S1-Bits für die Outputs 23 - 14
	5828 - 5891	UES für freie Benutzung (nicht PALCE22V10)
GAL26CV12		
	0000 - 6343	AND-array
	6344 - 6367	S0/S1-Bits für die Outputs 27 - 15
	6368 - 6431	UES für freie Benutzung

Tabelle 5-3: JEDEC Adressen für die GAL/PALCE22V10-Familie

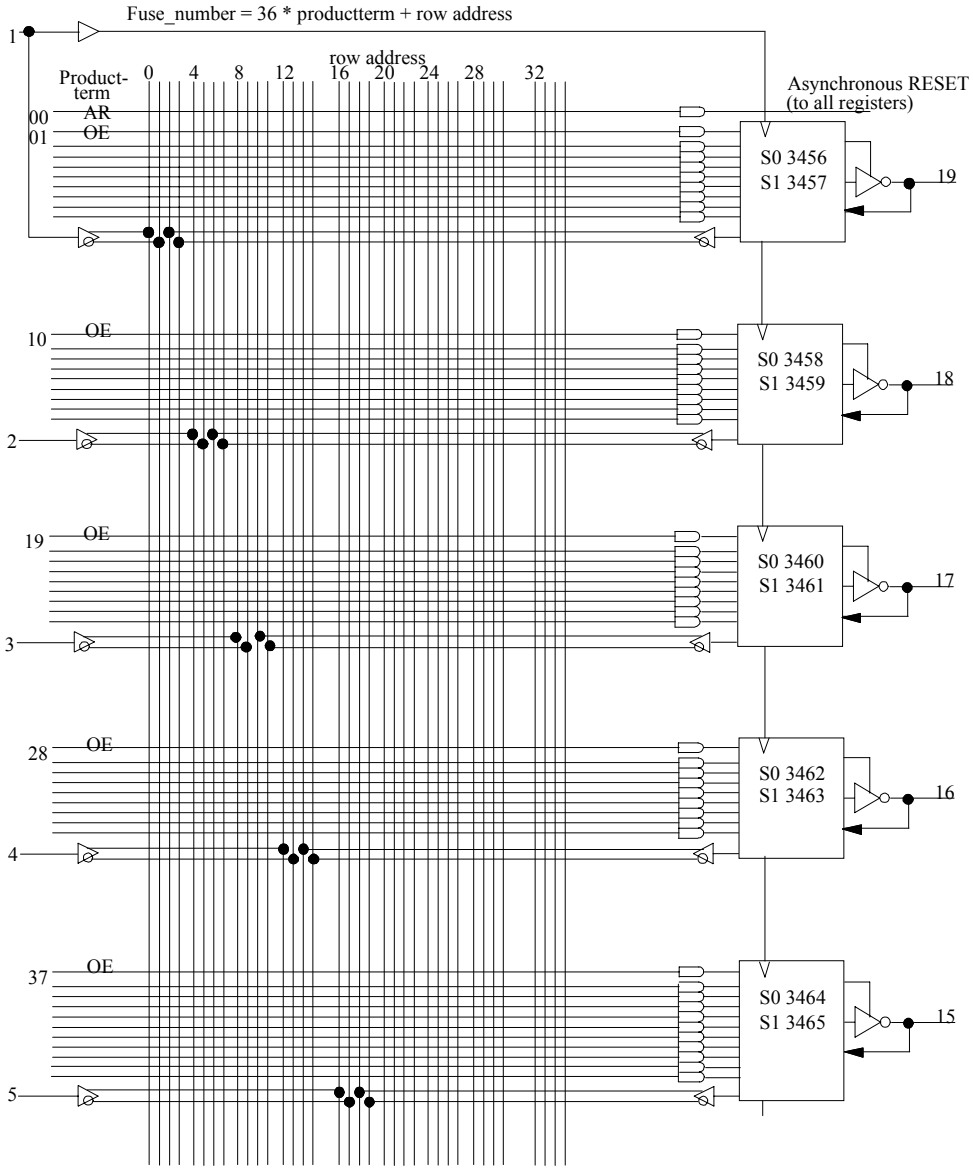


Abb.5-4a: Gal18V10 Fusemap Teil 1

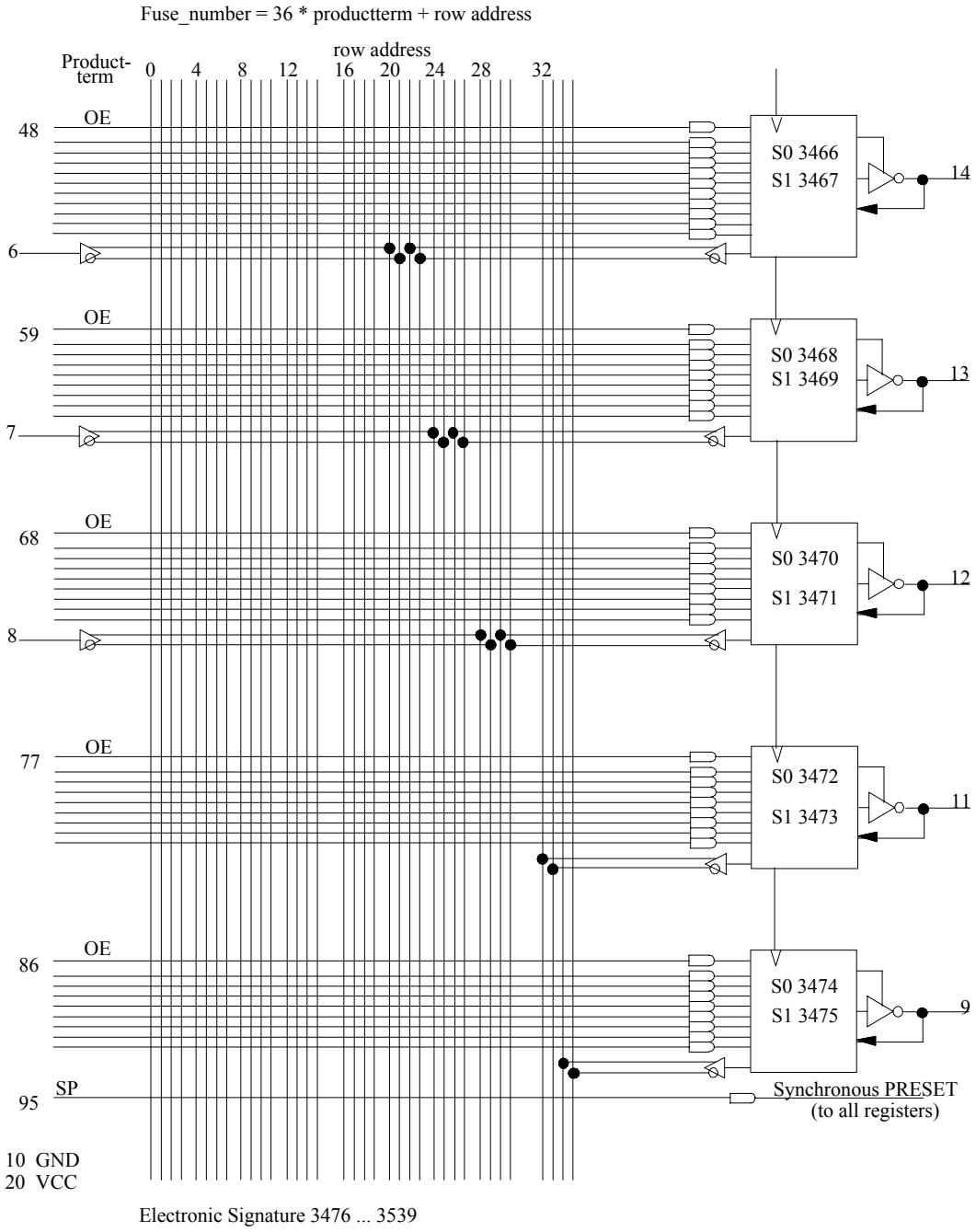


Abb.5-4b: GAL18V10 Fusemap Teil 2

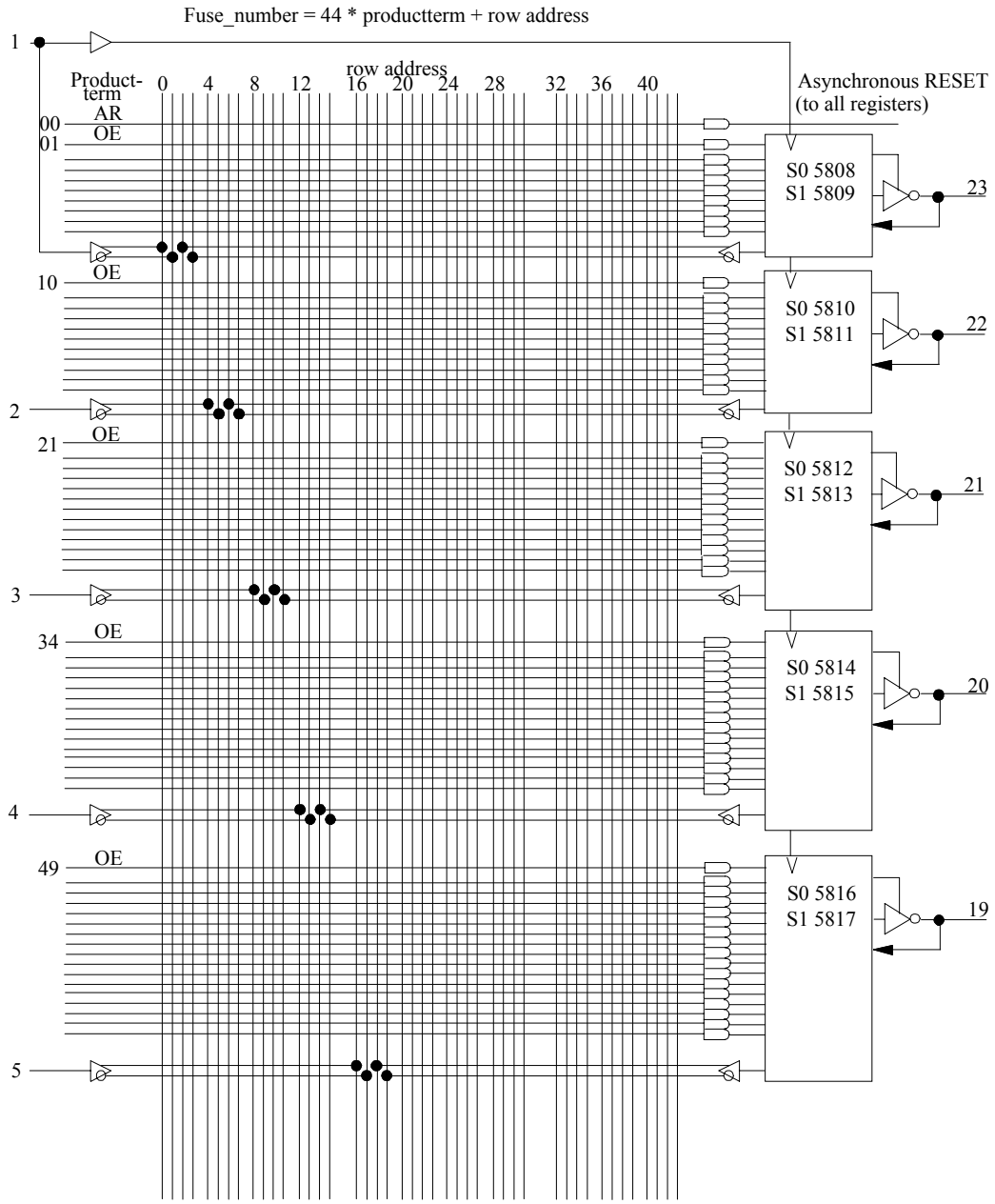


Abb.5-5a: GAL22V10 Fusemap Teil 1

Fuse_number = 44 * productterm + row address

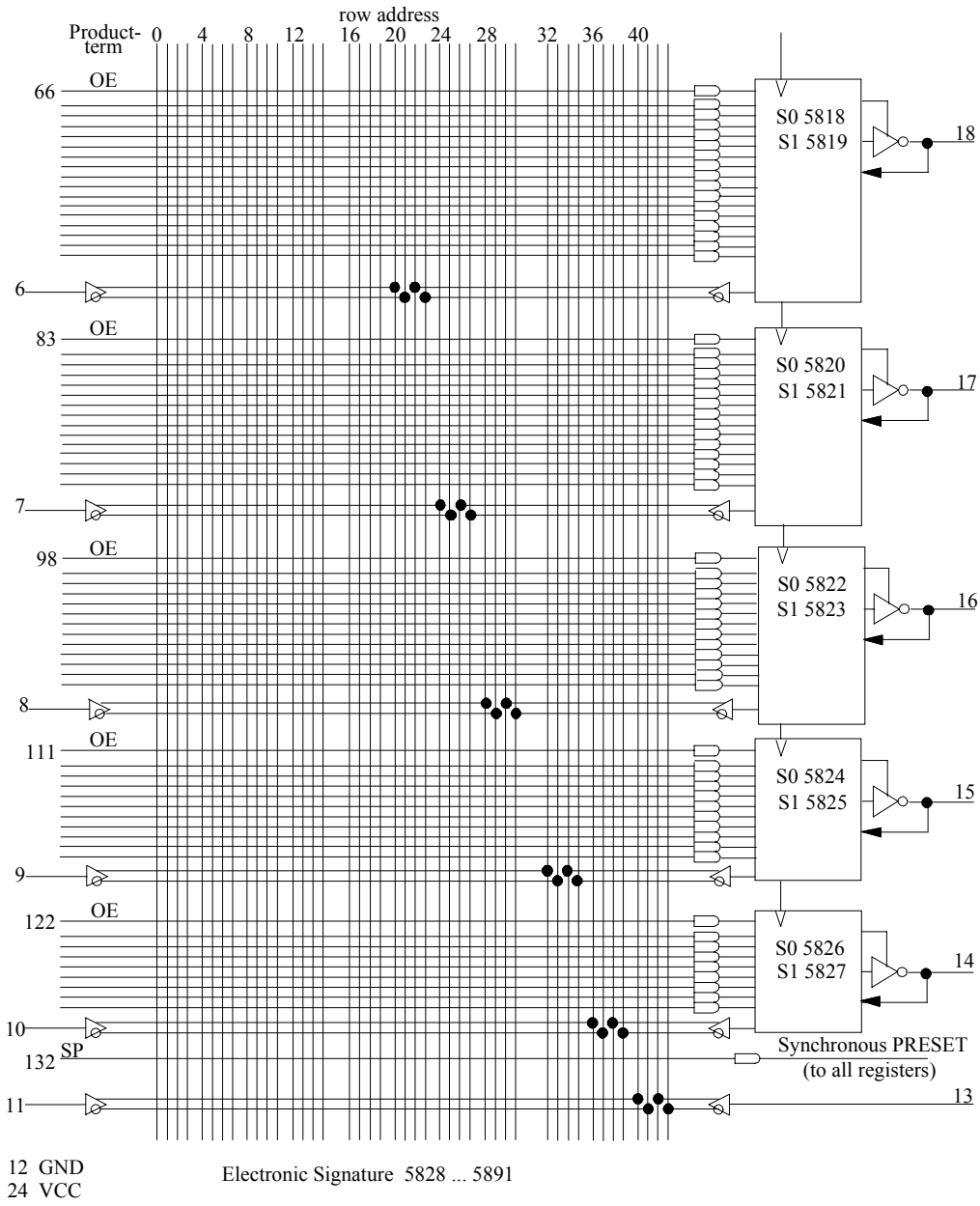


Abb.5-5b: GAL22V10 Fusemap Teil 2

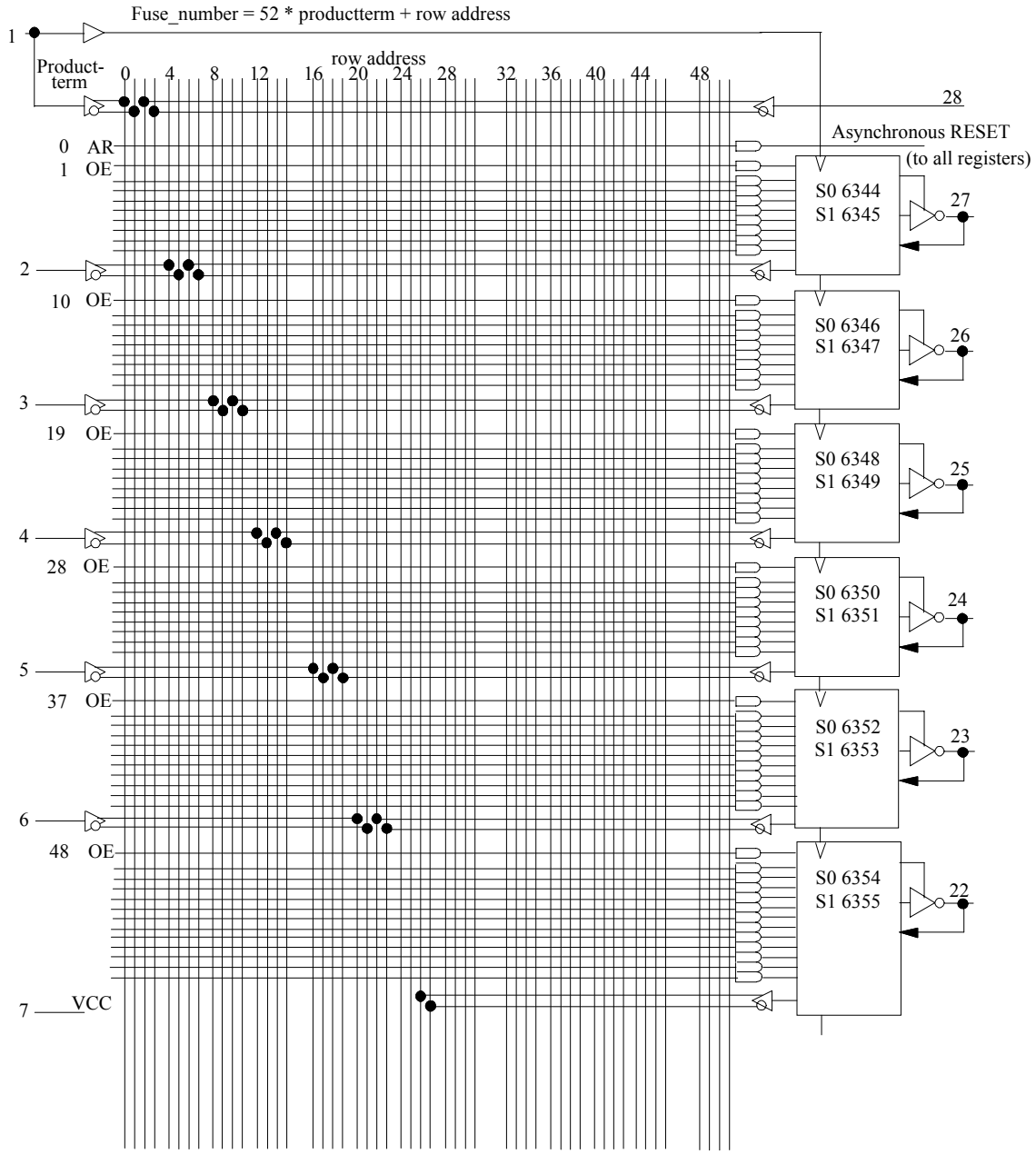


Abb.5-6a: GAL26CV12 Fusemap Teil 1

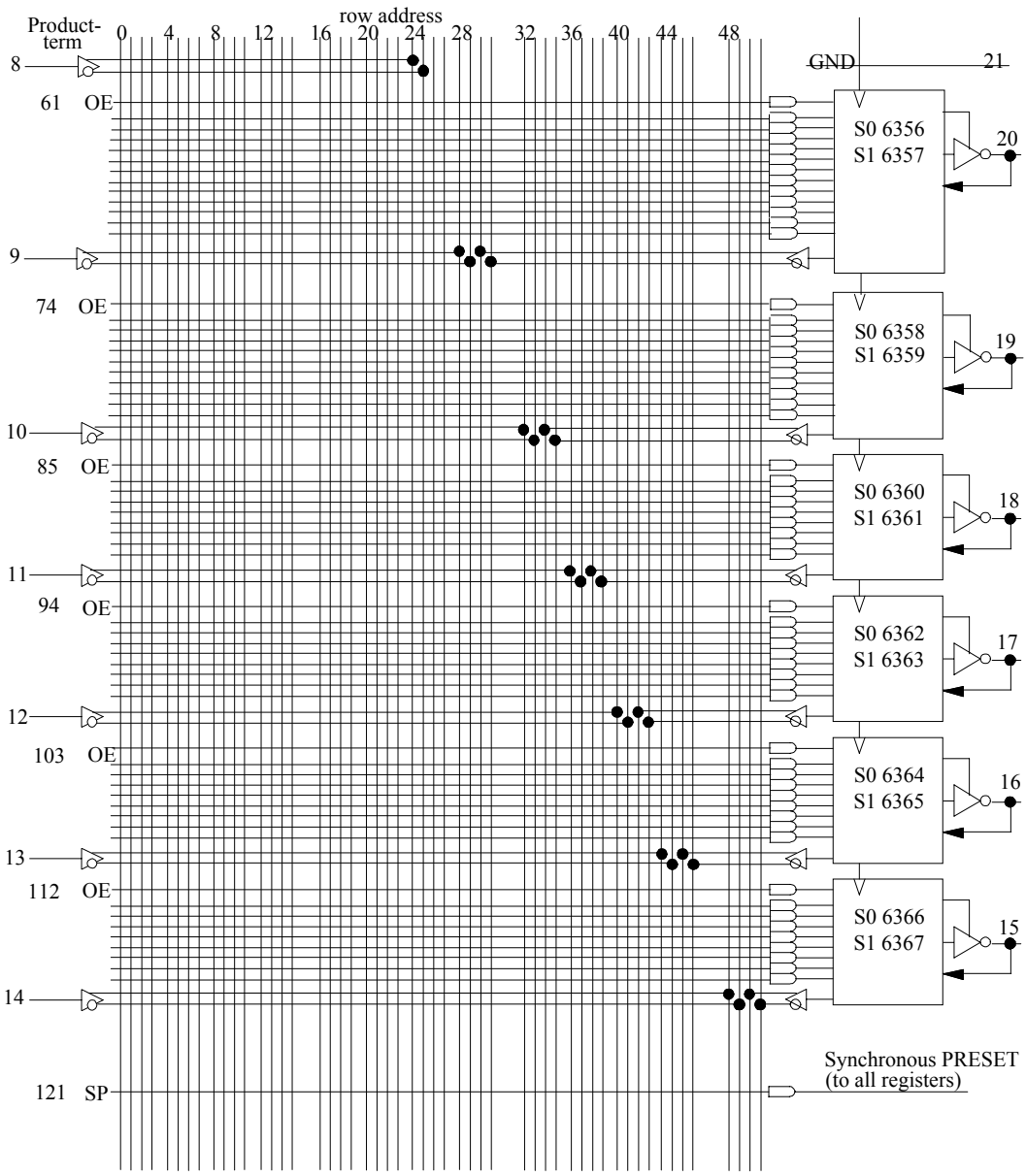


Abb.5-6b GAL26CV12 Fusemap Teil 2

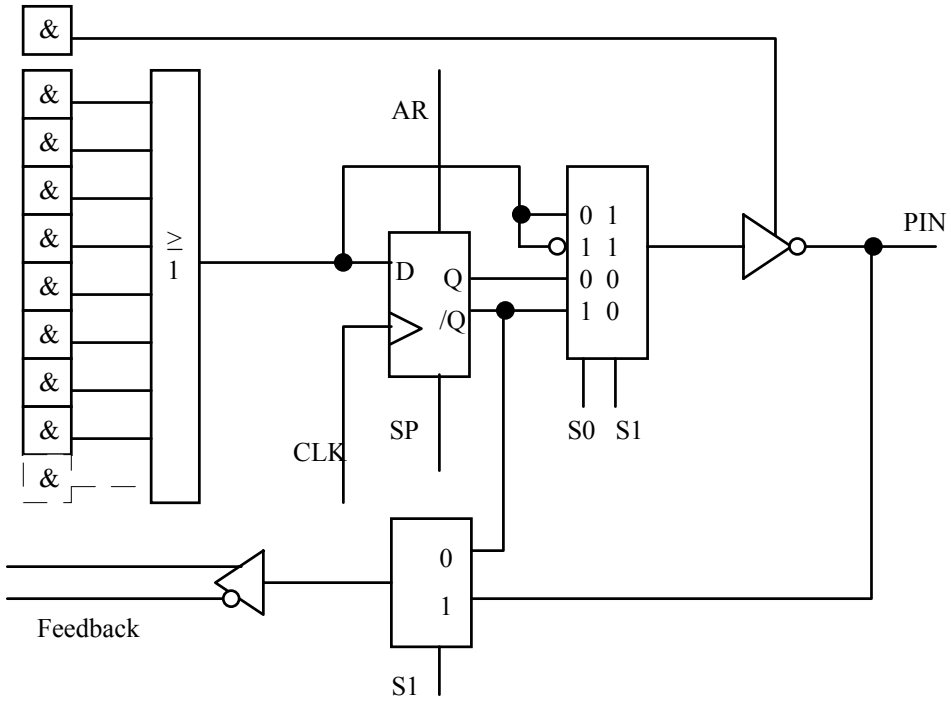


Abb.5-7: Output Logic Macrocell OLMC GAL 22V10

5.3 Die GAL20RA10-Familie

Der neueste GAL ist der GAL20RA10, der einerseits als eine Fortentwicklung der GAL 22V10-Familie angesehen werden kann, andererseits jedoch so neue Strukturen in sich trägt, daß er hier als neue Familie aufgeführt wird.

Die Weiterentwicklung geht in diesem Fall nicht in Richtung erhöhter Fuse-Anzahl oder erhöhter Komplexität, sondern in Richtung einer neuerlich veränderten Struktur, mit der der Nutzer Schaltungen integrieren kann, die bislang in den GAL-Familien nicht möglich waren. Gleichzeitig wird allerdings dadurch der Weg des General-Purpose-PLDs mit den bisherigen Familien zumindestens zum Teil verlassen, der GAL20RA10 gehört eher zu den Special-Purpose-PLDs.

Abb.5-8 zeigt die Fusemap des GALs 20RA10, Abb.5-9 die zugehörige OLMC. Der 20RA10 besteht aus 10 dedizierten Inputpins, 10 I/O-Pins sowie 2 speziellen Eingängen, /OE (Output Enable) und /PL (PreLoad), beide aktiv low. Die Inputpins sowie die I/O-Pins sind untereinander identisch. Zu jedem I/O-Pin, der sowohl synchron oder asynchron genutzt werden kann, gehören 8 Produktterme, von denen 4 zur jeweiligen logischen Verknüpfung genutzt werden können.

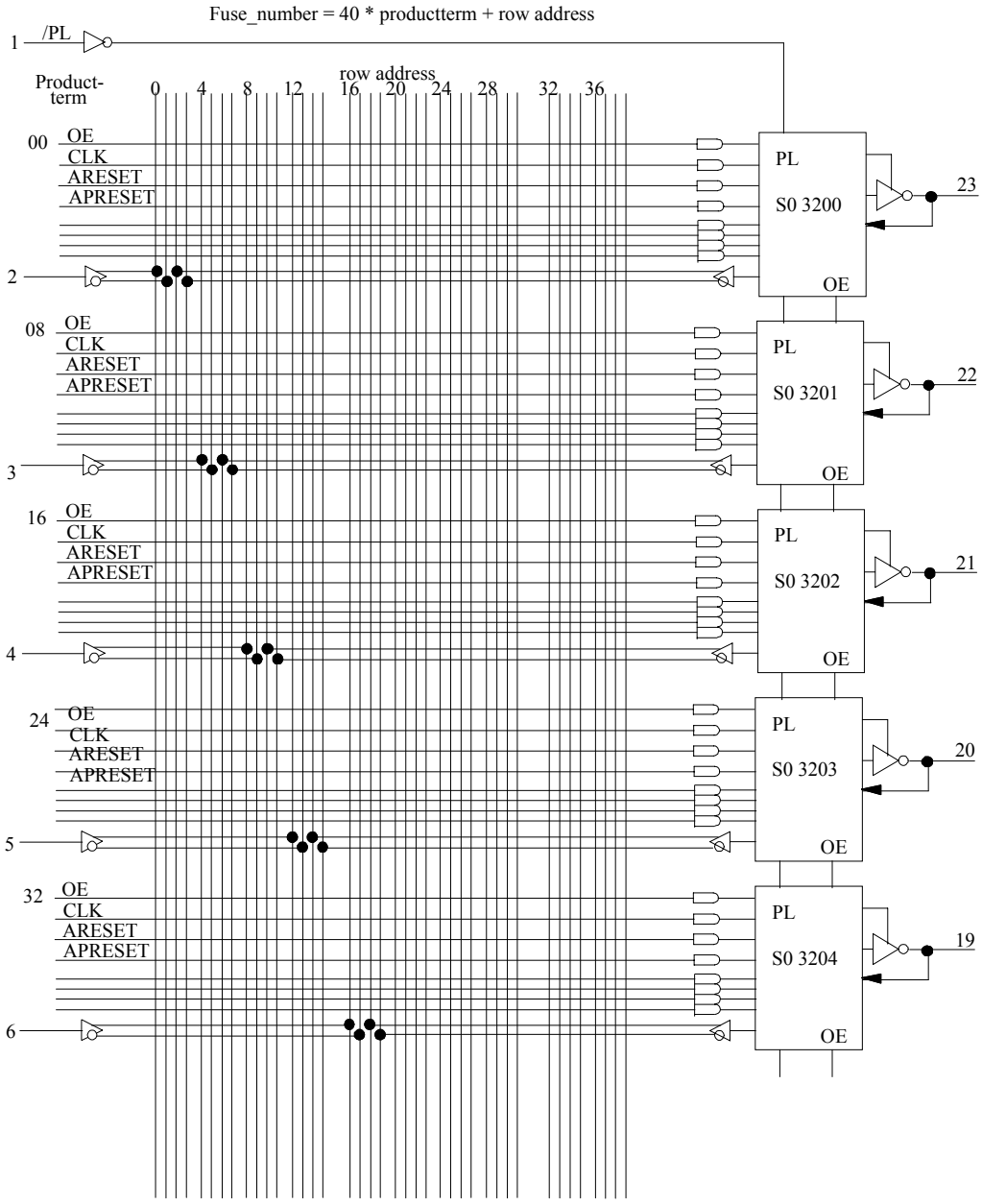


Abb.5-8a: GAL20RA10 Fusemap Teil 1

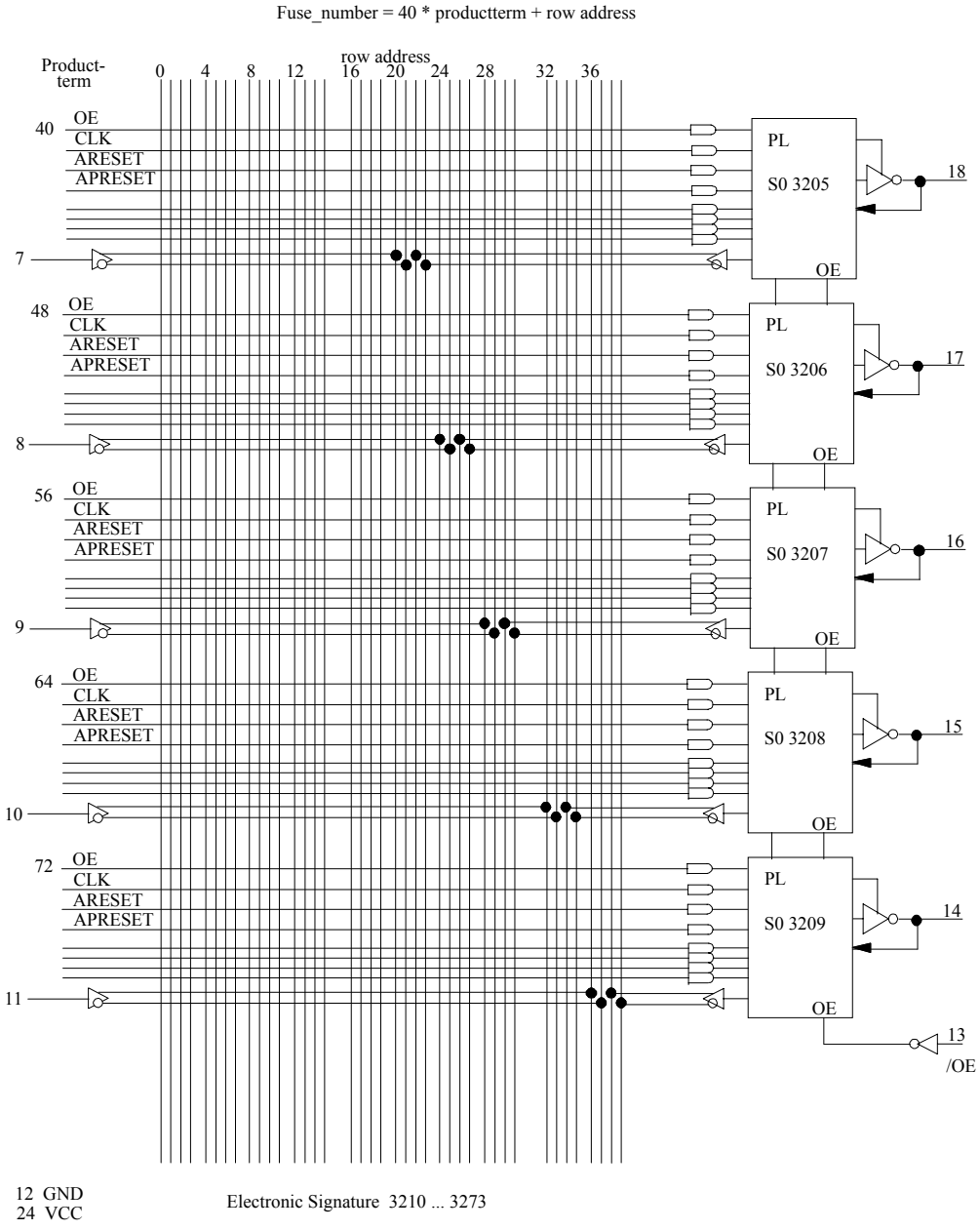


Abb.5-8b: GAL20RA10 Fusemap Teil 2

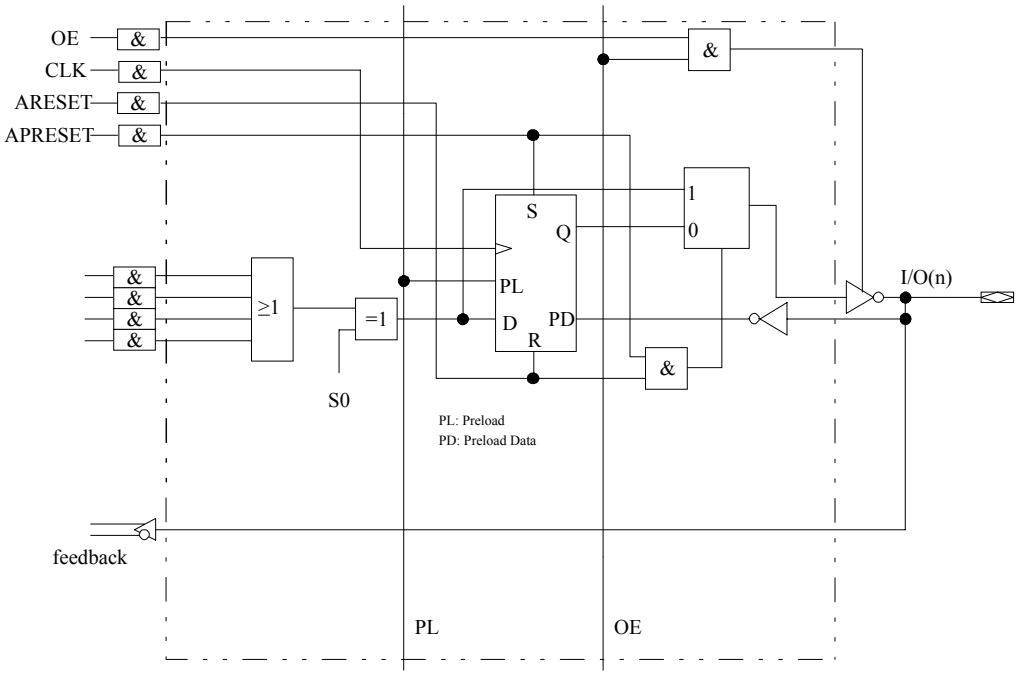


Abb.5-9: Output Logic Macrocell OLMC GAL20RA10

Die übrigen vier Produktterme haben spezielle Aufgaben, und hier liegt das Besondere im GAL 20RA10. Je ein Produktterm wird zur Tristate-Bedingung (diejenige für den Pin wird immer mit dem Eingang /OE AND-verknüpft), zur CLK-Bestimmung, zur Bestimmung von Asynchronous Reset und Asynchronous Preset bestimmt. Somit ist für jeden Register-Output eine Definition des Taktes notwendig, was aber gerade die Besonderheit dieses GALs ausmacht, denn jeder Takt kann einzeln bestimmt werden. CLK, Asynchroner Reset und -Preset sind natürlich für asynchrone Ouputs undefiniert; allerdings wird durch Asynchr. Reset und Preset, beide 'immer wahr', der Bypass am Register aktiv geschaltet, der asynchrone Output also gerade eingeschaltet, diese Aufgabe übernimmt GDS für Sie. Für die Register-Outputs, die auf Flanken an diesen Termen reagieren, wird dieser Zustand intern unterdrückt.

/OE sollte im Betrieb immer auf low gelegt sein, um die Outputpins überhaupt aktiv schalten zu können. /PL hingegen sollte inaktiv, also high im normalen Betrieb sein, da die Register nicht vorgeladen werden sollen. Die Ausnahme ist dann gegeben, wenn das Preloading aktiviert wird, dies geschieht im normalen Betrieb durch die Sequenz

- /OE high, um die Ausgangstreiber auszuschalten
- gewünschtes Datenwort an den I/O-Pins anlegen (Wartezeit mind. 10 us)
- Low-Impuls an /PL (Dauer mind. 35 ns).

Die Preload-Werte sind dann in den Registern gespeichert.

Tabelle 5-4 gibt Ihnen zum Abschluß den Überblick über die Fuse Adressen.

	Fuse Adresse	Funktion
GAL20RA10		
	0000 - 3199	AND-array
	3200 - 3209	S0-Bits für die Outputs 23 - 14
	3210 - 3273	UES für freie Benutzung

Tabelle 5-4: JEDEC Adressen für den GAL20RA10

5.4 Der ispGAL22V10

Auch der ispGAL22V10 entspricht in seiner internen Logik dem GAL22V10, die User-ID eingeschlossen. Von seiten der Logik sind demnach keine Umsetzungen vorzunehmen.

Die isp-Bausteine - isp steht hier für In-System-Programmable - benötigen jedoch zusätzliche Pins zur Programmierung, so daß die Pinzahl des ispGAL22V10 größer als die 24 Pins des GAL22V10 (DIL-Gehäuse) sein muß.

Der Hersteller Lattice wählte für den ispGAL22V10 ein 28poliges PLCC-Pinout, das gegenüber dem GAL22V10 im 28poligen PLCC im Betrieb pinkompatibel ist. Die 4 Pins, die dort mit NC bezeichnet sind, werden für die isp-Fähigkeit mit speziellen Funktionen belegt. Abb. 5-10 zeigt die Pinbelegungen für den ispGAL22V10 (PLCC) und den GAL22V10 (DIL).

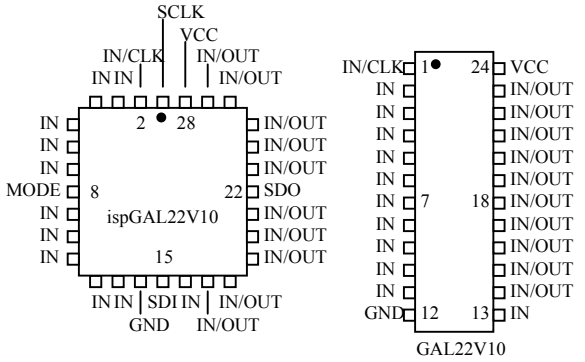


Abb. 5-10: Pinbelegungen ispGAL22V10, GAL22V10

Die Konsequenz für den Programmierer liegt in der Benennung der zusätzlichen Pins am ispGAL22V10, damit GDS den Sourcecode korrekt assemblieren kann. Die Pins heißen im einzelnen SCLK (Pin 1), MODE (Pin 8), SDI (Pin 15) und SDO (Pin 22). Diese Anschlüße entsprechen am PLCC-Gehäuse des 'normalen' GAL22V10 NC-Anschlüßen, sind also dort unbelegt. Während des normalen Betriebs sollten sie definiert auf Low gezogen werden.
Die Pinbelegung zum Interface des ispGALs22V10 und einem PC kann aus den Datenbücher von Lattice entnommen werden; es ist in das spezielle Kabel zum Anschluß von isp-Bausteinen integriert.

5.5 Die GDSxx-Familie

Als neue IC-Familie mit isp-Programmierungseigenschaften wurde die GDS-Familie eingeführt (GDS: Generic Digital Switch). Diese Bausteine beinhalten eine Verbindungsmatrix zwischen zwei Eingangsreihen, A und B, sowie konfigurierbaren Ausgangstreibern an jedem (!) Ein/Ausgang.

Die Grundidee der GDS-Bausteine ist der Ersatz von DIP-Switches, wobei dies nur insofern gelingt, daß eine Richtung ersatzweise geschaltet wird, während die Verbindungschaltung bei DIP-Switches bekanntermaßen richtungsunabhängig ist.

Abb. 5-11 zeigt die Pinbelegung der 3 verfügbaren GDS-Bausteine, Abb. 5-12 den Aufbau der I/O-Zelle, die für die Konfiguration eines Pins als Ein- oder Ausgang verantwortlich ist:

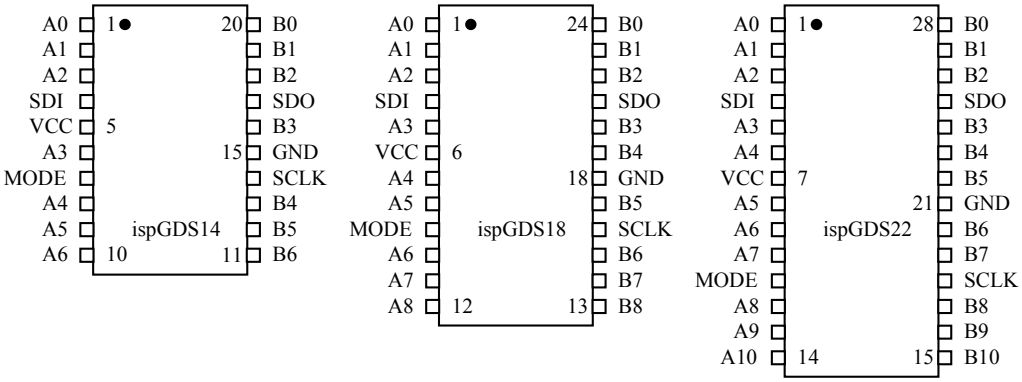


Abb. 5-11: Pinout der GDS-Bausteine

Die hier gezeigten DIP-Pinouts sind für den GDS14 auch als 20-Pin-PLCC, für den GDS22 als 28-Pin-PLCC erhältlich.

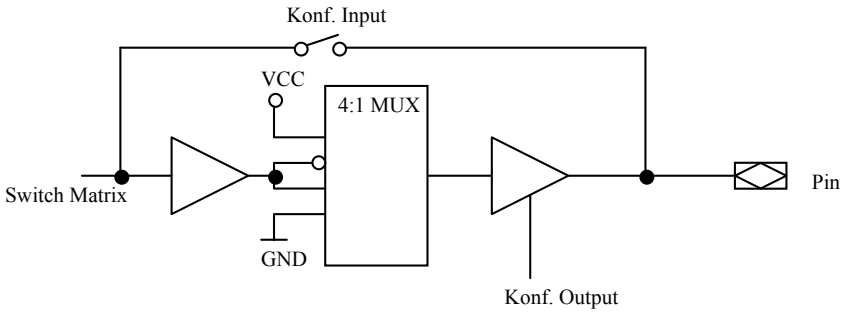


Abb. 5-12: I/O-Zelle der GDS-Bausteine

Innerhalb der GDS-Bausteine können A-Pins, die als Eingang konfiguriert sind (durch Sie!), an beliebige B-Pins, die als Ausgang genutzt werden, (normal oder invertiert) verschaltet werden; gleiches ist umgekehrt möglich, wobei jedem Ausgang auch ein fester TTL-Pegel (High oder Low, siehe Abb. 5-12, zugewiesen werden kann. Kopplungen Ax - Ax oder Bx - Bx sind jedoch nicht möglich.

Die GDS-Bausteine werden ebenfalls in isp-Technologie hergestellt und sind über das gleiche isp-Kabel programmierbar.

Anhang A: Fehler und Warnungsmeldungen

A1: Warnungsmeldungen

GDS 3.5 gibt einige Warnungsmeldungen aus, die - falls der Errorlevel auf 1 gesetzt ist - keine Unterbrechung des gesamten Assembliervorgangs bewirken, allerdings in der Fehlerliste vorhanden sind. Diese Warnungen sollten in jedem Fall beachtet werden, da hierdurch verborgene Fehler entstehen könnten, die im Rahmen einer Syntaxanalyse durch GDS 3.5 nicht feststellbar sind.

W001 Warnung: CLK und /OE sind nicht in demselben Sinn definiert! Zeile

Die GALs/PALCEs 16V8, 20V8 und 16Z8 benötigen eine Definition des Modus, in dem sie arbeiten sollen (synchron oder asynchron). Der synchrone Modus wird durch die beiden Pins 'CLK' (Pin 1) und '/OE' (Pin 11/13) automatisch erreicht, ansonsten wird der asynchrone Modus gewählt. Wird jetzt einer der beiden entsprechend definiert, der andere jedoch nicht, generiert GDS diese Warnungsmeldung und geht in den asynchronen (oder kombinatorischen) Modus. Zur Vermeidung deklarieren Sie bitte keinen oder beide Pins mit dem gewünschten Namen.

W002 Warnung: Wort/Marke zu lang für korrekte Assemblierung! Zeile

Die nutzbare Wortlänge für Pinnamen (ohne eventuell führendes '/') beträgt 14 , für Makronamen 19 Zeichen. Diese Warnung macht auf die Überschreitung aufmerksam, wobei während der Syntaxanalyse die maximal mögliche Anzahl der Zeichen als Identifizierer genutzt werden.

W003 Warnung: Definitionen sind zu lang! Zeile

Der Inhalt einer Textmakrodefinition ist auf 127 Zeichen begrenzt; die bezeichnete Makrodefinition überschreitet diese Grenze, so daß der weitere Teil fortgelassen werden mußte. Diese Warnung sollte sehr ernstgenommen werden, da hieraus ungewollte semantische Zusammenhänge geschaffen werden können, die keineswegs zu einem Syntaxfehler führen müssen.

W004 Warnung: Zuviele Definitionen! Zeile

Die Anzahl der Makrodefinitionen ist in GDS 3.5 auf 20 beschränkt. Weitere Anweisungen mittels #DEFINE müssen - ausgenommen die Redefinitionen, die keinen neuen Makronamen , sondern nur neuen Inhalt definieren - ignoriert werden.

W005 Warnung: Eventuell kein Feedback möglich! Zeile

In älteren GALs der 16V8-Familie (nicht bei PALCEs!) kann die Rückkopplung von Ausgängen im Simple Mode ggf. nicht möglich sein. GDS V3.5 weist auf diesen Umstand hin, der nicht weiter überprüft werden kann, aber im Betrieb sehr störend sein kann.

A2: Fehlermeldungen

In GDS 3.5 können mehrere Fehlermeldungen während eines Assembliervorganges generiert werden. Im Detail sind dies:

A001 Sourcecodezeile ist zu lang! Zeile

Eine Sourcecodezeile im precompilierten und minimierten Code, also ohne Kommentierung, jedoch mit Makroersetzung und Entklammerung, darf 2000 Zeichen nicht überschreiten. Bitte beachten Sie, daß eine Sourcecodezeile mehrere Textzeilen überstreichen kann und die komplette Definition des Zusammenhangs zwischen Outputpin und den Inputpins enthält. Die weitere Assemblierung dieser Zeile wird abgebrochen.

A002 Syntaxfehler in Zeile

Ein Syntaxfehler trat in der entsprechenden Zeile auf. Der Grund dafür kann z.B. in zwei aufeinanderfolgenden Pinnamen ohne Verknüpfung, unbekannten oder unmittelbar aufeinanderfolgenden Verknüpfungszeichen liegen. Die Klammerung unterliegt auch einigen Restriktionen, die Sie beachten müssen.

A003 Unbekannter Inputpin in Zeile

Ein nicht deklarierter Inputpin trat in der angegebenen Zeile auf.

A004 Kein Input möglich in Zeile

Es gibt einige Pins, an denen kein Input in den PLD möglich ist, dies aber versucht wurde. Dies tritt normalerweise auf, falls bei den GALs/PALCEs 16V8 und 20V8 im COMPLEX_MODE die Pins 12/19 bzw. 15/22, im SIMPLE_MODE die Pins 15/16 bzw. 19/20 als Input oder auch nur in der Rückkopplung in die AND-Matrix benutzt werden sollen, was nicht möglich ist. Hinweise finden Sie im Kapitel *DETAILS über GALs*.

A005 Unbekannter Outputpin in Zeile

Es wurde versucht, einen nicht deklarierten Pin als Outputpin zu definieren.

A006 Kein Outputpin in Zeile
Ein Pin, der tatsächlich in der Deklarationsliste vorhanden ist, aber durch die GAL-Architektur nicht als Outputpin benutzt werden kann, sollte als solcher definiert werden.

A007 Zweifach definierter Outputpin in Zeile

Es wurde versucht, einen Outputpin zweimal zu definieren.

A008 Kombinatorischer Output vorgeschrieben in Zeile

An dieser Stelle ist die Benutzung eines synchronen (Register-) Outputs nicht möglich, sondern es muß kombinatorisch definiert werden. Die Ursachen können in der Modus-Wahl (kombinatorisch, GALs 16V8, 20V8 und 16Z8) oder darin liegen, daß z.B. interne Pins nur asynchron zu nutzen sind (z.B. AR und SP in den GALs 18V10, 22V10 und 26CV12). Die GDS-Bausteine lassen lediglich kombinatorische Ausgangszuweisungen zu.

A009 Kein Tristate möglich in Zeile

Ein sollte eine Tristate-Bedingung für einen Outputpin definiert werden, wo dies nicht möglich ist. Der Grund kann z.B. bei den GALs 16V8, 20V8 und 16Z8 sein, daß für synchrone Outputs die Tristate-Kontrolle immer durch den Pin /OE (Pin 11/13) geschieht oder der SIMPLE_ MODE gewählt wurde; für die Mitglieder der GAL22V10-Familie gibt es weiterhin keine Tristate-Möglichkeiten für interne Pins. Gleiches gilt für alle Ausgänge der GDS-Bausteine.

A010 Keine OR-Zeilen mehr verfügbar in Zeile

Die Anzahl der Produktterme (alle Pins, die nur mit '*' verknüpft sind, gehören zu einem Produktterm) ist begrenzt durch die interne Architektur der jeweiligen GALs. Diese Terme werden durch '+' verbunden, und es wurde versucht, mehr Terme als vorhanden zu verbinden.

In der GAL22V10-Familie variiert die Anzahl der Produktterme von Outputpin zu Outputpin, so daß durch Pinwechsel eventuell eine Abhilfe möglich ist. GDS-Bausteine haben generell keine Verknüpfungsmöglichkeiten, sondern dienen lediglich als Schalter.

A011 Keine AND-Zeilen mehr verfügbar in Zeile

AND-Verknüpfungen sind in normalen GALs in dem Maße vorhanden, daß diese Meldung nicht vorkommen kann. Eine Ausnahme bilden PLAs (zukünftige Erweiterung!) und GDS-Bausteine, wo nur begrenzte AND-Gatter integriert sind, dafür aber die ODER-Gatter programmierbar sind.

A012 Kein gültiges CHIP-Kommando ! Zeile

Es wurde kein 'CHIP'-Kommando ('CHIP' am Zeilenanfang in großen Buchstaben!) im Sourcecode gefunden. Bitte korrigieren Sie dies und assemblieren Sie erneut.

A013 Nicht genügend Pins definiert ! Zeile

Ein Gleichheitszeichen (= oder :=) wurde gefunden, bevor die Pinliste vollständig deklariert war, so daß die Pinanzahl nicht mit den Vorgaben übereinstimmt. Bitte korrigieren Sie dies, wobei Sie beachten sollten, daß interne Pins wie AR und SP in der GAL22V10-Familie automatisch deklariert werden.

A014 Unbekannter GAL-Typ ! Zeile

Der deklarierte GAL-Typ ist unbekannt.

A015 Groundpin nicht oder falsch definiert ! Zeile

Die Deklaration des Groundpins muß immer an der vorgesehenen Stelle erfolgen.

A016 VCC Pin nicht oder falsch definiert ! Zeile

Ebenso muß die Deklaration des VCC-Pins an der richtigen Stelle in der Pinliste erfolgen.

A017 ispGALs: SCLK, SDI, SDO oder MODE nicht oder falsch definiert! Zeile

Die speziellen Pins SCLK, SDI, SDO und MODE müssen deklariert werden, falls einer der ispGALs benutzt werden soll. Diese Deklaration wird durch den Assembler überprüft, und es wurde mindestens einer dieser Pins mit einem Fehler gefunden. Bitte korrigieren Sie die Deklaration anhand der Detailinformationen zu den GALs.

A018 Outputnegation nicht erlaubt ! Zeile

Die internen 'Pins' der GAL22V10-Familie (18V10, 22V10 und 26CV12) können nicht invertiert werden, dies wurde jedoch versucht. Bitte korrigieren Sie unter Beachtung der Verknüpfungslogik die Definition dieser Pins.

A019 Kein Asynchronous Reset Extension möglich in Zeile

Bei dem GAL20RA10 können - neben der Tristate-Definition - auch Asynchronous Reset, Preset und Clock für Outputs definiert werden. Dies war an dieser Stelle nicht möglich.

A020 Kein Asynchronous Preset Extension möglich in Zeile

Wie A019

A021 Kein Clockangabe möglich in Zeile

Wie A019

A022: Keine Pullupangabe möglich in Zeile

(Zukünftige Erweiterung)

A023 Keine Feedbackangabe möglich in Zeile

(Zukünftige Erweiterung)

A024: Keine Power-Up-Resetangabe möglich in Zeile

(Zukünftige Erweiterung)

A025 GAL20RA10: /PL und/oder /OE nicht oder falsch definiert ! Zeile

Für den GAL20RA10 müssen die Pins 1 und 13 als /PL (PreLoad) bzw. /OE (Output Enable) definiert werden. Dies wurde fehlerhaft vorgenommen.

A026 Keine Clockdefinition für Pin ...! Zeile

Ein Pin im GAL20RA10, der synchron definiert wird, braucht grundsätzlich eine Definition seines Clock-Inputs. Dies wurde vergessen. Bei dieser Fehlermeldung werden die Pinnummer und die Zeilennummer der Pin-Definition angegeben.

A027 Dieser Eingabepin für Clock ist nicht zulässig! Zeile

(Zukünftige Erweiterung)

A028 GDSxx-IC: UND-Logik nicht gestattet! Zeile

Bei allen GDS-Bausteinen ist keine Kombinatorik möglich, lediglich den Ausgängen kann ein Eingangspin oder ein fester TTL-Wert zugeordnet werden.

Sie haben versucht, UND-Verknüpfungen für GDS-Bausteine einzufügen.

A029 Unerwartetes Dateiende in Zeile

Die Sourcecodedatei wurde für die Syntaxanalyse unerwartet beendet. Dies kann z.B. auftreten, wenn eine Blockkommentierung mittels /* .. begonnen, jedoch nicht beendet wurde.

A030 Unerwartete Klammer in Zeile

Sie benutzen eine schließende Klammer mehr als Sie öffnende Klammer in der zugehörigen Sourcecodezeile gesetzt haben, oder sie steht an zu früher Stelle.

A031 Klammer erwartet in Zeile

Sie benutzen in der zugehörigen Sourcecodezeile mindestens eine öffnende Klammer mehr als schließende Klammern, so daß der Assembler eine weitere Klammer erwartet.

A032 GAL Definitionsfile nicht lesbar!

Der Definitionsfile für die Assemblierung, GDS_G35D.DEF, konnte unter diesem Namen nicht zum Lesen geöffnet werden, so daß die Assemblierung abgebrochen werden mußte. Bitte überprüfen Sie Ihre Installation von GDS.

A033 Unbekannter Fehler in Zeile

Es wurde ein Syntaxfehler im Sourcecode entdeckt, ohne daß GDS diesen in eine der bisherigen Kategorien einordnen konnte. Bitte wenden Sie sich mit diesem Fehler an SH-Elektronik, Marthastr. 8, 24114 Kiel.

Anhang B: Sourcecode der Beispielfiles

B1: DTACK68.GAL

Diese Datei enthaelt den Sourcecode fuer einen Waitstate-Generator fuer ein 68000-System. Der Generator, basierend auf einem GAL22V10, kreiert ein DTACK Signal fuer die CPU mit 0 bis 3 Waitstates, abhaengig von der Adresslage. Die maximale Aufloesung im Adressraum ist 64 kByte, so daß also fuer jeweils
64 kByte ein einheitliches Signal generiert werden muss, das aber von Block zu Block dann variieren kann.

Wie ueblich beginnt der Sourcecode mit dem Kommando

```
CHIP   WAIT_STATE_68000     GAL22V10

CLK_IN A23    A22    A21    A20    A19    A18    A17    A16  R_W    DS     GND
FC0    FC1    FC2    /FC0_1_2 WAIT0  WAIT1  DTACK  WAIT2  WAIT3  AS     DTACK_EXT
VCC

; Nunmehr sind die Pins deklariert, die Logikgleichungen beginnen
; (Bitte beachten Sie, dass dies Kommentarzeilen sein muessen)

AR     =      /DS          ; DS ('Data Select') wird als externe NAND-Kombina-
                           ; tion von UDS und LDS angenommen. AR ist ein inter-
                           ; nes Signal im GAL22V10 fuer Asynchronen Reset

WAIT0  :=     /WAIT0 ; Fuer DS low wird WAIT0 immer zurueckgesetzt (durch
                           ; den asynchronen Reset AR!), bei DS high toggled
                           ; WAIT0 mit CLK_IN/2. WAIT0 wird speziell fuer
                           ; Lesezugriffe der 68000-CPU generiert, da die Dauer
                           ; von DS zwischen Lesen und Schreiben differiert

WAIT1  :=      /WAIT1 * /R_W
               + WAIT1 * /R_W
               + WAIT0 * R_W     ;

WAIT2  :=     WAIT1          ; 2 wait states

WAIT3  :=     WAIT2          ; 3 wait states
; WAIT0 bis WAIT3 sind zwar Outputsignale, aber nicht direkt durch die CPU
; genutzt, da diese ein /DTACK-Signal benoetigt (asynchrones Busprotokoll der
; 68000-CPU). Dieses Signal wird nunmehr aus den Waitstate-Signalen und den
; Adressleitungen generiert.
/DTACK :=     /A23 * DS * /FC0_1_2    ; niedrigsten 4 MWord 0 zero waitstates
               + A23 * A22 * A21 * A20 * DS * /FC0_1_2 * WAIT3
                        ; oberstes MByte 3 waitstates
               + A23 * /A22 * /A21 * A20 * /A19 * /A18 * /A17 * /A16
               * DS * /FC0_1_2 * DTACK_EXT
                        ; externes DTACK in $80xxxx
; Dies hier generierte DTACK-Signal soll nur als Beispiel dienen, wie flexibel ein DTACK-
Generator mit
; Hilfe eines GALs programmiert werden kann. Das FC0_1_2-Signal (function code) wird in die
AND-Matrix
; nicht-; invertiert rueckgekoppelt (im Vergleich mit der Pindeklaration!), da das Signal 'low'
benoetigt wird,
; wenn die ; Logikgleichung das Resultat 'high' ergibt. Fuer weitere Informationen zur
Rueckkopplung von
; Signalen siehe ; Kapitel 4

FC0_1_2 =     FC0 * FC1 * FC2 * /AS   ; Hilfssignal fuer Funktionscode
```

B2: ERROR1.GAL

Diese Beispieldatei enthaelt den Sourcecode eines GALs zur Adressdekodierung einer PC-Interfaceplatine.

Auf diesem Board sind AD- und DA-Wandler sowie ein PIO-Adapter integriert. Absichtlich wurden einige Fehler im Sourcecode integriert, die beim Assemblieren durch GDS detektiert werden sollten.

Das Board benoetigt einen Adressbereich von insgesamt 32 Adressen im I/O-Bereich. Innerhalb dieses Bereichs selektieren die Chip-Select-Leitungen wie folgt:

- Basisadresse (BA) bis BA + 1: /CS0 aktiv (low)
- BA + 4 bis BA + 7: /CS1 aktiv
- BA + 16 bis BA + 23: /CSA aktiv
- BA + 24 bis BA + 31: /CSB aktiv
- Alle anderen Adressen bewirken keinen Selektierung
- /CS als Summensignal ist aktiv low, falls ein /CSx aktiv ist

```
CHIP    Universal_IO_mod      GAL16V8           COMPLEX_MODE


A0      A1      A2      A3      A4      IORD    IOWR    CS16    NC      GND
NC      CS      B_IOWR  B_IORD  CSB     CSA     CS1     NC      CS0     VCC

/CS0    :=      !CS16 & !A4 & !A3 * !A2 & !A1 & !IORD
                | /CS16 * /A4 * /A3 * /A2 * /A1 * /IOWR    ;       BA bis BA+1

                ; gemischte Anwendung der zugelassenen logischen Verknüpfungen.

/CS1    =       /CS16 * /A5 * /A3 * A2 * /IORD
                + /CS16 * /A4 * /A3 * A2 * /IOWR ;              BA+4 bis BA+7

/CSA    =       /CS16 * A4 * /A3 * /IORD
                + /CS16 * A4 * /A3 * /IOWR ;                    BA+16 bis BA+23

/CSB    =       /CS16 * A4 * A3 / /IORD
                + /CS16 * A4 * A3 * /IOWR ;                     BA+24 bis BA+31

CS      =       CS0 * CS1 * CSA * CSB ;                         Summensignal

B_IORD  =               IORD ;

B_IOWR  =               IOWR ;                                  gepufferte Ausgaenge

USER_ID =               TEST1           ; hier ein Beisspiel für den Eintrag der User-Id
                                        ; Schauen Sie sich auch den JEDEC-Code an !!
```

B3: UNIVERS.GAL

(fehlerfreie Version von ERROR1.GAL)

Diese Beispieldatei enthaelt den Sourcecode eines GALs zur Adressdekodierung einer PC-Interfaceplatine. Auf diesem Board sind AD- und DA-Wandler sowie ein PIO-Adapter integriert.

Das Board benoetigt einen Adressbereich von insgesamt 32 Adressen im I/O-Bereich. Innerhalb dieses Bereichs selektieren die Chip-Select-Leitungen wie folgt:

- Basisadresse (BA) bis BA + 1: /CS0 aktiv (low)
- BA + 4 bis BA + 7: /CS1 aktiv
- BA + 16 bis BA + 23: /CSA aktiv
- BA + 24 bis BA + 31: /CSB aktiv
- Alle anderen Adressen bewirken keinen Selektierung
- /CS als Summensignal ist aktiv low, falls ein /CSx aktiv ist

```
CHIP    Universal_IO_mod      GAL16V8           COMPLEX_MODE

A0      A1      A2      A3      A4      IORD    IOWR    CS16    NC      GND
NC      CS      B_IOWR  B_IORD  CSB     CSA     CS1     CS0     NC      VCC

/CS0    =       !CS16 & !A4 & !A3 * !A2 & !A1 & !IORD
                | /CS16 * /A4 * /A3 * /A2 * /A1 * /IOWR    ;       BA bis BA+1

                ; gemischte Anwendung der zugelassenen logischen Verknüpfungen.

/CS1    =       /CS16 * /A4 * /A3 * A2 * /IORD
                + /CS16 * /A4 * /A3 * A2 * /IOWR ;              BA+4 bis BA+7

/CSA    =       /CS16 * A4 * /A3 * /IORD
                + /CS16 * A4 * /A3 * /IOWR ;                    BA+16 bis BA+23

/CSB    =       /CS16 * A4 * A3 * /IORD
                + /CS16 * A4 * A3 * /IOWR ;                     BA+24 bis BA+31

CS      =       CS0 * CS1 * CSA * CSB ;                         Summensignal

B_IORD  =               IORD ;

B_IOWR  =               IOWR ;                                  gepufferte Ausgaenge

USER_ID                 TEST1           ; hier ein Beisspiel für den Eintrag der User-Id
                                        ; Schauen Sie sich auch den JEDEC-Code an !!
```

B4: TT20RA10GAL

Diese Beispieldatei dient als Beispiel zur Ausnutzung der vielfaeltigen Moeglichkeiten des GALs 20RA10. Es werden Ausgaenge definiert, die sowohl synchron als auch asynchron mit verschiedenen Sonderbedingungen ausgestattet sind.

Letzte Aenderung: 30.09.1997

```
CHIP    Example_20RA10  GAL20RA10

/PL     CLOCK1 /CE     A1   A2   A3   CLOCK2 A4   A5  A_RESET A_PRESET    GND
/OE     /CS    CS0     CS1  CS2  CS3  NC     NC         NC      NC     NC  VCC

; Bei der Deklaration muessen Pin 1 als /PL (PreLoad) und Pin 13 als /OE
; (Output Enable) deklariert werden.

; Jetzt beginnen die Gleichungen fuer diesen GAL, die nur Beispielcharakter  haben

/CS0            :=      /A5 * /A4 * /A3 * /A2
CS0.CLCK        =       CLOCK1 ;                Diese Bedingung ist notwendig
CS0.TRST        =       CE ;                    Tristate und Asynchr. Reset sind optional
CS0.ARST        =       A_RESET ;

/CS1            :=      /A5 * /A4 * A3 * /A2
CS1.CLCK        =       CLOCK1 * CLOCK2 ;       AND-Kombination fuer Clock ist beim
                                        ;       GAL20RA10 moeglich
CS1.APRST       =       A_PRESET        ;       Optional, Tristate hier implizit immer
                                        ;       wahr

/CS2            :=      /A5 * A4 * /A3 * /A2 * A1
CS2.CLCK        =       CLOCK2 ;
CS2.TRST        =       CE ;
CS2.ARST        =       A_RESET * CLOCK1        ; Auch hier AND-Kombination moeglich
CS2.APRST       =       A_PRESET ;

/CS3            :=      A5 * A4 + A5 * /A4 * /A3 * /A2
                        + A5 * /A4 * A3 * A2 + A5 * /A4 * A3 * /A2 * A1
                                ; Beim GAL20RA10 sind allerdings nur 4 Produktterme pro
                                ; Outputpin möglich
CS3.CLCK        =       CLOCK1 ;
CS3.TRST        =       CE ;

CS              =       /CS0 + /CS1 + /CS2 + /CS3 ;
                                ; CS ist low, wenn einer der Ausgaenge low ist ! Bitte
                                ; ueberpruefen Sie dies, denn hier wird rueckgekoppelt !
```

B5: IMPULS01.GAL

Dieses Beispiel dient der Demonstration der Flexibilitaet von PLDs allgemein und von GALs im speziellen.

Impulsschaltungen sind allgemein alle Schaltungen, die in geeigneter Art und Weise zeitlich veraenderliche Signale produzieren. Beispiele dafuer sind Rechteckgeneratoren mit bestimmten Zeitverhaeltnissen, Nadelimpulsgeneratoren, Schmitt-Trigger sowie Verzoegerungsschaltungen.

Dieses Beispielprogramm wurde fuer den GAL22V10 geschrieben; sicherlich wird ein solcher GAL sehr selten Eingang in Schaltungen finden, wohl aber die Einzelkomponenten, die auch in GALs eingebaut werden koennen, falls noch gerade "etwas Platz" dort zu finden war.

Die hier aufgefuehrten Beispiele wurden einem Artikel der Zeitschrift Elektronik 20/1991, Seite 112ff entnoommen, dem auch detaillierter Angaben zu den Zeitkonstanten zu entnehmen sind.

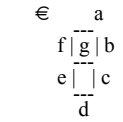
```
CHIP    IMPULS01        GAL22V10

NC     R_ECK_IN     NC      NADEL_IN     NCTRIG_FF     RUECK_FF     NC
SCHMITT_IN     NC        NC      GND
NC  NC    NC   SCHMITT_OUT    NC  HELP_FF    MONO_FF     NADEL_2    NADEL_1
R_OUT_2R_OUT_1   VCC

; Als erste Schaltung wird der Rechteckgenerator vorgestellt:
; Das folgende "Bild" zeigt die schematische externe Beschaltung

;       |-------------------------------------------------------|
;       |----R_ECK_IN                           R_OUT_1---|     |
;                                                      ---      |
;                                                R     | |      |
;                                                      ---      |
;                                                       |------|
;                                                       |
;                                                     -----
;                                                C    -----
;                                             R_OUT_2---|

; R und C bestimmen die Zeitkonstante und somit die Rechteckdauer Die beiden
; Ausgaenge muessen immer gegensaetzliche Potentiale annehmen. Durch diesen Trick
; und der Rueckkopplung des Potentials zwischen R und  C kommt es zu Ladung und
; Entladung des Kondensators, also einem zeitlich Aufeinanderfolgen der Zustaende
; HIGH und LOW an jedem der; beiden Ausgaenge!

R_OUT_1         =       /R_ECK_IN       ; beide Ausgaenge sind immer aktiv!

R_OUT_2         =       /R_OUT_1        ;

;*****************************************************************************
; Die zweite Schaltung aehnelt der erstgenannten, wobei nunmehr ein; Nadelimpuls
; erzeugt werden soll:. Die externe Beschaltung hat folgende Gestalt:
;                                                          VCC
;       |----NADEL_IN                           NADEL_1--    |
;       |                                                   ---
;       |                                             R     | |
;       |                                                   ---
;       |                                       NADEL_2----|------|
;       |-------------------------------------------------|      -----
;                                                          C    -----
;                                                               |
;                                                     GND       -----

; Durch den hochohmigen Zustand von NADEL_2 in der Phase, waehrend NADEL_1
; den Wert LOW hat, wird der Kondensator geladen, bis der Wert an NADEL_IN
```

```
; die Schaltschwelle errreicht hat und der Kondensator schlagartig entladen  wird.
;
; Auch hier ist der Abstand der Nadelimpulse ueber die Zeitkonstante RC festgelegt
;
; Der Nadelimpuls kann an NADEL_1 abgegriffen werden.


NADEL_1             =       NADEL_IN ;

NADEL_2             =       NADEL_IN  *  /NADEL_IN ;
NADEL_2.TRST        =       NADEL_1 ;

; Die Definition fuer NADEL_2 ergibt staendig ein LOW-Wert fuer die lo-
; gische Verknuepfung, wobei dieser Wert nur fuer NADEL_1 = HIGH auf dem
; Pin anliegt, ansonsten ist dieser im High-Impedance-Zustand.

;*************************************************************************
; Die dritte Applikation besteht in einem monostabilen Flipflop:
;                                                          | VCC
;                                                          |------|
;                                                         ---    ---
;                                                      R | |   D /|\
;                                                        ---     ---
;                                                          |------|
;                                                          |------|
;                                                         -----    |
;       -----TRIG_FF                          MONO_FF--   ----- C  |
;       |----RUECK_FF                         HELP_FF-------|      |
;       |-----------------------------------------------------------|


/HELP_FF     =      TRIG_FF  +  /RUECK_FF ;
MONO_FF      =      /HELP_FF ;

; Die Erklaerung der Schaltung: Mit TRIG_FF HIGH wird HELP_FF auf LOW
; geschaltet. Danach bleibt - unabhaengig von TRIG_FF - HELP_FF auf LOW-
; Pegel, bis der Kondensator soweit geladen ist, dass an RUECK_FF die
; Schwellspannung ueberschritten wird und der Pegel HIGH angenommen  wird.

; Ist nunmehr TRIG_FF wiederum auf LOW, so schaltet MONO_FF in den LOW-
; Pegel zurueck. Die eingebaute Diode ermoeglicht dabei eine rasche  Entladung des
Kondensators.


;*************************************************************************
; Das letzte Beispiel dieses Sourcefiles stellt eine einfache Variante des Schmitt-
Triggers dar.


;                        |---------------|-------|-------------------------|
;                                        -------  R2                        |
;       _____ R1         |                                                  |
;   --|_____|---|----SCHMITT_IN                    SCHMITT_OUT----|
;

SCHMITT_OUT  =      SCHMITT_IN ;

; R1 und R2 fungieren dabei als Spannungsteiler zwischen der Eingangsspannung
; an R1 und der Ausgangsspannung an R2.
```

B6: SIEBSEG.GAL

Dieses GAL-Programm dient zum Betreiben einer invertiert angesteuerten 7-Segment Anzeige,d.h. das jeweilige Segment leuchtet, wenn der Relais-kontakt geöffnet ist. Es erscheinen nacheinander die Ziffern 0 bis 9, gesteuert durch das Taktsignal vom Timer. Der jeweils nächste Zustand wird aus dem vorherigen generiert. (Statemachine)

Ziffer	Segment a b c d e f g
0	0 0 0 0 0 0 1
1	1 0 0 1 1 1 1
2	0 0 1 0 0 1 0
3	0 0 0 0 1 1 0
4	1 0 0 1 1 0 0
5	0 1 0 0 1 0 0
6	0 1 0 0 0 0 0
7	0 0 0 1 1 1 1
8	0 0 0 0 0 0 0
9	0 0 0 0 1 0 0



Die Zustände der Ausgänge der einzelnen Ziffern werden zunächst als Makros definiert. Dadurch wird der Programmtext übersichtlicher, gleichzeitig wird Schreibarbeit eingespart. Vor dem Assemblieren ist ein Precompiler-Lauf erforderlich.Dieser erzeugt eine Datei mit der Endung ".pre",in der die Makros durch die wahren Gleichungen ersetzt wurden. Diese Datei kann nun in das JEDEC-Format übersetzt werden.

```
CHIP    SIEBENSEG     GAL16V8        COMPLEX_MODE

;Makrodefinitionen
#define NULL   = /a * /b * /c * /d * /e * /f * g
#define EINS   = a * /b * /c * d * e *  f * g
#define ZWEI   = /a * /b */c * /d * /e * f * /g
#define DREI   = /a * /b * /c * /d * e * f * g
#define VIER   = a * /b * /c * d * e * /f * /g
#define FÜNF   = /a * b * /c * /d * e* /f * /g
#define SECHS  = /a * b * /c * /d * /e * /f * /g
#define SIEBEN = /a * /b * /c * d * e * f * g
#define ACHT   = /a * /b * /c * /d * /e * /f * /g
#define NEUN   = /a * /b * /c * /d * e * /f * /g

CLK   NC   NC   NC   NC   NC   NC   NC   NC   GND
/OE   SOE   g     f     e     d     c     b     a     VCC

a  := NULL        ; a setzen im Anschluß an Ziffer "0"
     + DREI       ; und Ziffer "3"
b  := VIER        ; b setzen im Anschluß an Ziffer "4"
   + FÜNF         ; und Ziffer "5"
c  := EINS        ; c setzen im Anschluß an Ziffer "1"
d  := NULL        ; d setzen im Anschluß an Ziffer "0"
   + DREI         ; und Ziffer "3"
   + SECHS         ; und Ziffer "6"
/e := EINS        ; e zurücksetzen im Anschluß an Ziffer "1"
   + FÜNF         ; und Ziffer "5"
   + SIEBEN        ; und Ziffer "7"
   + NEUN         ; und Ziffer "9"
f  := NULL        ; f setzen im Anschluß an Ziffer "0"
   + EINS         ; und Ziffer "1"
   + ZWEI          ; und Ziffer "2"
   + SECHS         ; und Ziffer "6"
g  := NULL        ; g setzen im Anschluß an Ziffer "0"
   + SECHS        ; und Ziffer "6"
   + NEUN          ; und Ziffer "9"

SOE := GND      ; Synchron-Output Enable : Am Anfang alle Ausgänge auf 0
```

B7a: ZAEHL_1:GAL

Diese Schaltungsbeschreibung zeigt die Programmierung eines Binaerzaehlers mit einer Breite von 10 Bit. Der Zaehlvorgang wird durch einem COUNT_EN-Eingang gesteuert, der nur bei LOW ein Weiterzaehlen gestattet. Ein LOW-Zustand an RESET (asynchron) setzt alle Zaehlerbits auf 0.

Diese 1. Version wird ohne Klammerung beschrieben.

```
CHIP    ZAEHL_1                 GAL22V10

CLK     NC      /RESET NC       NC      COUNT_EN        NC      NC  NC  NC      NC      GND
NC      D0      D2      D4      D6      D8      D9      D7      D5  D3          D1      VCC

AR      =       RESET  ; Asynchroner Reset

D0      :=       /D0 * /COUNT_EN + D0 * COUNT_EN

D1      :=       /D1 * D0 * /COUNT_EN + D1 * /D0 * /COUNT_EN
               + D1 * COUNT_EN

D2      :=       /D2 * D1 * D0 * /COUNT_EN
               + D2 * /D1 * /COUNT_EN + D2 * /D0 * /COUNT_EN
               + D2 * COUNT_EN

D3      :=       /D3 * D2 * D1 * D0 * /COUNT_EN
               + D3 * /D2 * /COUNT_EN + D3 * /D1 * /COUNT_EN
               + D3 * /D0 * /COUNT_EN
               + D3 * COUNT_EN

D4      :=       /D4 * D3 * D2 * D1 * D0 * /COUNT_EN
               + D4 * /D3 * /COUNT_EN + D4 * /D2 * /COUNT_EN
               + D4 * /D1 * /COUNT_EN + D4 * /D0 * /COUNT_EN
               + D4 * COUNT_EN

D5      :=       /D5 * D4 * D3 * D2 * D1 * D0 * /COUNT_EN
               + D5 * /D4 * /COUNT_EN + D5 * /D3 * /COUNT_EN
               + D5 * /D2 * /COUNT_EN + D5 * /D1 * /COUNT_EN
               + D5 * /D0 * /COUNT_EN
               + D5 * COUNT_EN

D6      :=       /D6 * D5 * D4 * D3 * D2 * D1 * D0 * /COUNT_EN
               + D6 * /D5 * /COUNT_EN + D6 * /D4 * /COUNT_EN
               + D6 * /D3 * /COUNT_EN + D6 * /D2 * /COUNT_EN
               + D6 * /D1 * /COUNT_EN + D6 * /D0 * /COUNT_EN
               + D6 * COUNT_EN

D7      :=       /D7 * D6 * D5 * D4 * D3 * D2 * D1 * D0 * /COUNT_EN
               + D7 * /D6 * /COUNT_EN + D7 * /D5 * /COUNT_EN
               + D7 * /D4 * /COUNT_EN + D7 * /D3 * /COUNT_EN
               + D7 * /D2 * /COUNT_EN + D7 * /D1 * /COUNT_EN
               + D7 * /D0 * /COUNT_EN
               + D7 * COUNT_EN

D8      :=       /D8 * D7 * D6 * D5 * D4 * D3 * D2 * D1 * D0 * /COUNT_EN
               + D8 * /D7 * /COUNT_EN + D8 * /D6 * /COUNT_EN
               + D8 * /D5 * /COUNT_EN + D8 * /D4 * /COUNT_EN
               + D8 * /D3 * /COUNT_EN + D8 * /D2 * /COUNT_EN
               + D8 * /D1 * /COUNT_EN + D8 * /D0 * /COUNT_EN
               + D8 * COUNT_EN

D9      :=       /D9 * D8 * D7 * D6 * D5 * D4 * D3 * D2 * D1
                  * D0 * /COUNT_EN
               + D9 * /D8 * /COUNT_EN + D9 * /D7 * /COUNT_EN
               + D9 * /D6 * /COUNT_EN + D9 * /D5 * /COUNT_EN
               + D9 * /D4 * /COUNT_EN + D9 * /D3 * /COUNT_EN
               + D9 * /D2 * /COUNT_EN + D9 * /D1 * /COUNT_EN
               + D9 * /D0 * /COUNT_EN
               + D9 * COUNT_EN
```

B7b: ZAEHL_2.GAL

Diese Schaltungsbeschreibung zeigt die Programmierung eines Binaerzaehlersmit einer Breite von 10 Bit. Der Zaehlvorgang wird durch einem COUNT_EN-Eingang gesteuert, der nur bei LOW ein Weiterzaehlen gestattet.

Ein LOW-Zustand an RESET (asynchron) setzt alle Zaehlerbits auf 0.

Diese 2. Version wird mit Klammerung (ab GDS V3.5) beschrieben.

```
CHIP    ZAEHL_2                 GAL22V10

CLK     NC      /RESET NC       NC      COUNT_EN        NC      NC  NC  NC      NC      GND
NC      D0      D2      D4      D6      D8      D9      D7      D5  D3          D1      VCC

AR      =       RESET  ; Asynchroner Reset

D0      :=       /D0 * /COUNT_EN + D0 * COUNT_EN

D1      :=       (/D1 * D0 + D1 * /D0) * /COUNT_EN
               + D1 * COUNT_EN

D2      :=       (/D2 * D1 * D0 + D2 * (/D1 + /D0)) * /COUNT_EN
               + D2 * COUNT_EN

D3      :=       (/D3 * D2 * D1 * D0 + D3 * (/D2 + /D1 + /D0)) * /COUNT_EN
               + D3 * COUNT_EN

D4      :=       (/D4 * D3 * D2 * D1 * D0
               + D4 * (/D3 + /D2 + /D1 + /D0)) * /COUNT_EN
               + D4 * COUNT_EN

D5      :=       (/D5 * D4 * D3 * D2 * D1 * D0
               + D5 * (/D4 + /D3 + /D2 + /D1 + /D0)) * /COUNT_EN
               + D5 * COUNT_EN

D6      :=       (/D6 * D5 * D4 * D3 * D2 * D1 * D0
               + D6 * (/D5 + /D4 + /D3 + /D2 + /D1 + /D0)) * /COUNT_EN
               + D6 * COUNT_EN

D7      :=       (/D7 * D6 * D5 * D4 * D3 * D2 * D1 * D0
               + D7 * (/D6 + /D5 + /D4 + /D3 + /D2 + /D1 + /D0))
                        * /COUNT_EN
               + D7 * COUNT_EN

D8      :=       (/D8 * D7 * D6 * D5 * D4 * D3 * D2 * D1 * D0
               + D8 * (/D7 + /D6 + /D5 + /D4 + /D3 + /D2 + /D1 + /D0))
                        * /COUNT_EN
               + D8 * COUNT_EN

D9      :=       (/D9 * D8 * D7 * D6 * D5 * D4 * D3 * D2 * D1 * D0
               + D9 * (/D8 + /D7 + /D6 + /D5 + /D4 + /D3 + /D2 + /D1 + /D0))
                        * /COUNT_EN
               + D9 * COUNT_EN
```

Anhang C: Überblick der Hotkeys und Editorkommandos

F10 **Menüleiste** F1 **Hilfe**

ALT+D **Datei**

Öffnen	F3
Sichern	F2
Beenden	ALT + X

ALT+B **Bearbeiten**

Ausschneiden	Shift + Delete
Kopieren	Ctrl + Insert
Einfügen	Shift + Insert
Löschen	Ctrl + Delete
Weitersuchen	Ctrl + L

ALT+C **Compiler**

Nur Syntaxcheck	F9
Assemblieren	ALT + F9
Nächster Fehler	ALT + F8
Vorheriger Fehler	ALT + F7

ALT+S **Simulation**

ALT+P **Programmieren**

ALT+O **Optionen**

ALT+F **Fenster**

Größe/Bewegen	CTRL + F5
Zoom	F5
Nächstes Fenster	F6
Voriges Fenster	ALT + F3

Anhang D: Übersicht der programmierbaren GALs

GDS Version 3.5 erlaubt die Programmierung mittels GDSProg und GDSProg2 für folgende GAL-Typen und Hersteller:

16V8:	Hersteller Lattice	Algorithmen 00 - 05
	Hersteller NSC	Algorithmen 00 - 05
20V8:	Hersteller Lattice	Algorithmen 00 - 05
	Hersteller NSC	Algorithmen 00 - 05
18V10:	Hersteller Lattice	Algorithmen 00 - 05
22V10:	Hersteller Lattice	Algorithmen 00 - 05
	Hersteller NSC	Algorithmen 00 - 01
26CV12:	Hersteller Lattice	Algorithmen 00 - 05
20RA10:	Hersteller Lattice	Algorithmen 00 - 05
	Hersteller NSC	Algorithmen 00 - 01
PALCE16V8:	Hersteller AMD	Algorithmus 00 - 01
PALCE22V10:	Hersteller AMD	Algorithmus 00 - 02 (Einschränkung bei GDSProg)
ispGAL22V10:	Hersteller Lattice	Algorithmus 00
GDS14, 18, 22:	Hersteller Lattice	Algorithmus 00

Alle übrigen GALs/PALCEs werden vor dem Programmier- oder Löschvorgang erkannt; der Algorithmus wird dann mit einer Fehlermeldung abgebrochen. Dies kann insbesondere bei speziellen Zero-Power-GALs/PALCEs der Fall sein!

Wenden Sie sich bitte an uns, falls ein GAL-Typ nicht erkannt oder programmiert werden kann.

Die Typen der Familien GAL 16V8 und 20V8 (A, B, D, AS usw.) werden automatisch vom GDS erkannt.

Hinweise, Ergänzungen, Sonstiges

Sollten Sie vor unüberwindlichen Schwierigkeiten stehen, Fehler entdeckt haben oder wollen Sie Verbesserungsvorschläge loswerden , wenden Sie sich an

SH-Elektronik
Jütlandring 41

D-24109 Kiel

Tel. 0431 / 66 51 16 Fax. 0431 / 67 41 09

email: info@sh-elektronik.de
web: www.sh-elektronik.de

Wir helfen Ihnen gerne weiter.